



Taking Touch to the Next Level.

Touch-enabled app development requires developers to re-think the future of their user experiences. Whether you're a WinForms developer building client applications or a Web developer building for mobile platforms, DevExpress 12.1 tools help you take on these new challenges using your existing skills & technologies available today.



Your users are ready.
Ensure you're ready, too.

Download your
free 30-day trial at
DevExpress.com

DXv2

The next generation of inspiring tools. **Today.**



msdn[®] magazine

VISUAL STUDIO 2012

A More Productive IDE for Modern Applications

Peter Vogel 34

What's New for Web Development in Visual Studio 2012

Clark Sell 42

Developing and Deploying Windows Azure Cloud Services Using Visual Studio

Boris Scholl and Paul Yuknewicz 48

What's New in Microsoft Test Manager 2012

Sudheer Adimulam, Micheal Learned and Tim Star 60

Testing for Continuous Development

Larry Brader and Alan Cameron Wills 66

Shape up Your Data with Visual Studio LightSwitch 2012

Jan Van der Haegen 72

COLUMNS

CUTTING EDGE

Mobile Site Development, Part 4: Managing Device Profiles
Dino Esposito, page 6

WINDOWS WITH C++

The Pursuit of Efficient and Composable Asynchronous Systems
Kenny Kerr, page 14

DATA POINTS

Moving Existing Projects to EF 5
Julie Lerman, page 22

FORECAST: CLOUDY

Humongous Windows Azure
Joseph Fultz, page 28

TEST RUN

Coding Logistic Regression with Newton-Raphson
James McCaffrey, page 78

TOUCH AND GO

Exploring Spherical Coordinates on Windows Phone
Charles Petzold, page 84

DON'T GET ME STARTED

On Honor, Cold Iron and Hot Silicon
David Platt, page 88

Start a Revolution

Refuse to choose between desktop and mobile.



With the brand new NetAdvantage for .NET,
you can create awesome apps with killer data
visualization today, on any platform or device.

Get your free, fully supported trial today!

www.infragistics.com/NET



Infragistics Sales US 800 231 8588 • Europe +44 (0) 800 298 9055 • India +91 80 4151 8042 • APAC (+61) 3 9982 4545

Copyright 1996-2012 Infragistics, Inc. All rights reserved. Infragistics and NetAdvantage are registered trademarks of Infragistics, Inc. The Infragistics logo is a trademark of Infragistics, Inc. All other trademarks or registered trademarks are the respective property of their owners.



PopulationExplosion

VISUALIZING THE GROWTH OF THE WORLD COMMUNITY

XamPivotGrid

XamDataChart

Country

Population

GDP Per

Measures

Albania

Austria

Belarus

Belgium

Bosnia and Herzegovina

Bulgaria

Croatia

Denmark

Estonia

Finland

France

Germany

Greece

Hungary

Iceland

Ireland

Italy

Latvia

Lithuania

Luxembourg

Macedonia, FYR

Moldova

Montenegro

Netherlands

Population

GDP Per

Measures

Albania

Austria

Belarus

Belgium

Bosnia and Herzegovina

Bulgaria

Croatia

Denmark

Estonia

Finland

France

Germany

Greece

Hungary

Iceland

Ireland

Italy

Latvia

Lithuania

Luxembourg

Macedonia, FYR

Moldova

Montenegro

Netherlands

Population

GDP Per

Measures

Albania

Austria

Belarus

Belgium

Bosnia and Herzegovina

Bulgaria

Croatia

Denmark

Estonia

Finland

France

Germany

Greece

Hungary

Iceland

Ireland

Italy

Latvia

Lithuania

Luxembourg

Macedonia, FYR

Moldova

Montenegro

Netherlands

Population

GDP Per

Measures

Albania

Austria

Belarus

Belgium

Bosnia and Herzegovina

Bulgaria

Croatia

Denmark

Estonia

Finland

France

Germany

Greece

Hungary

Iceland

Ireland

Italy

Latvia

Lithuania

Luxembourg

Macedonia, FYR

Moldova

Montenegro

Netherlands

Population

GDP Per

Measures

Albania

Austria

Belarus

Belgium

Bosnia and Herzegovina

Bulgaria

Croatia

Denmark

Estonia

Finland

France

Germany

Greece

Hungary

Iceland

Ireland

Italy

Latvia

Lithuania

Luxembourg

Macedonia, FYR

Moldova

Montenegro

Netherlands

Population

GDP Per

Measures

Albania

Austria

Belarus

Belgium

Bosnia and Herzegovina

Bulgaria

Croatia

Denmark

Estonia

Finland

France

Germany

Greece

Hungary

Iceland

Ireland

Italy

Latvia

Lithuania

Luxembourg

Macedonia, FYR

Moldova

Montenegro

Netherlands

Population

GDP Per

Measures

Albania

Austria

Belarus

Belgium

Bosnia and Herzegovina

Bulgaria

Croatia

Denmark

Estonia

Finland

France

Germany

Greece

Hungary

Iceland

Ireland

Italy

Latvia

Lithuania

Luxembourg

Macedonia, FYR

Moldova

Montenegro

Netherlands

Population

GDP Per

Measures

Albania

Austria

Belarus

Belgium

Bosnia and Herzegovina

Bulgaria

Croatia

Denmark

Estonia

Finland

France

Germany

Greece

Hungary

Iceland

Ireland

Italy

Latvia

Lithuania

Luxembourg

Macedonia, FYR

Moldova

Montenegro

Netherlands

Population

GDP Per

Measures

Albania

Austria

Belarus

Belgium

Bosnia and Herzegovina

Bulgaria

Croatia

Denmark

Estonia

Finland

France

Germany

Greece

Hungary

Iceland

Ireland

Italy

Latvia

Lithuania

Luxembourg

Macedonia, FYR

Moldova

Montenegro

Netherlands

Population

GDP Per

Measures

Albania

Austria

Belarus

Belgium

Bosnia and Herzegovina

Bulgaria

Croatia

Denmark

Estonia

Finland

France

Germany

Greece

Hungary

Iceland

Ireland

Italy

Latvia

Lithuania

Luxembourg

Macedonia, FYR

Moldova

Montenegro

Netherlands

Population

GDP Per

Measures

Albania

Austria

Belarus

Belgium

Bosnia and Herzegovina

Bulgaria

Croatia

Denmark

Estonia

Finland

France

Germany

Greece

Hungary

Iceland

Ireland

Italy

Latvia

Lithuania

Luxembourg

Macedonia, FYR

Moldova

Montenegro

Netherlands

Population

GDP Per

Measures

Albania

Austria

Belarus

Belgium

Bosnia and Herzegovina

Bulgaria

Croatia

Denmark

Estonia

Finland

France

Germany

Greece

Hungary

Iceland

Ireland

Italy

Latvia

Lithuania

Luxembourg

Macedonia, FYR

Moldova

Montenegro

Netherlands

Population

GDP Per

Measures

Albania

Austria

Belarus



dtSearch®

Instantly Search Terabytes of Text

- 25+ fielded and full-text search types
- dtSearch's **own document filters** support "Office," PDF, HTML, XML, ZIP, emails (with nested attachments), and many other file types
- Supports databases as well as static and dynamic websites
- **Highlights hits** in all of the above
- APIs for .NET, Java, C++, SQL, etc.
- 64-bit and 32-bit; Win and Linux

"lightning fast" Redmond Magazine

"covers all data sources" eWeek

"results in less than a second" InfoWorld

hundreds more reviews and developer case studies at www.dtsearch.com

dtSearch products:

- ◆ Desktop with Spider
- ◆ Web with Spider
- ◆ Network with Spider
- ◆ Engine for Win & .NET
- ◆ Publish (portable media)
- ◆ Engine for Linux
- ◆ Document filters also available for separate licensing

Ask about fully-functional evaluations

The Smart Choice for Text Retrieval® since 1991

www.dtSearch.com 1-800-IT-FINDS

msdn®

magazine

SEPTEMBER 2012 VOLUME 27 NUMBER 9

MITCH RATCLIFFE Director

MOHAMMAD AL-SABT Editorial Director/mmeditor@microsoft.com

PATRICK O'NEILL Site Manager

MICHAEL DESMOND Editor in Chief/mmeditor@microsoft.com

DAVID RAMEL Technical Editor

SHARON TERDEMAN Features Editor

WENDY HERNANDEZ Group Managing Editor

KATRINA CARRASCO Associate Managing Editor

SCOTT SHULTZ Creative Director

JOSHUA GOULD Art Director

CONTRIBUTING EDITORS Dino Esposito, Joseph Fultz, Kenny Kerr, Julie Lerman, Dr. James McCaffrey, Ted Neward, John Papa, Charles Petzold, David S. Platt

Redmond Media Group

Henry Allain President, Redmond Media Group

Doug Barney Vice President, New Content Initiatives

Michele Imgrund Sr. Director of Marketing & Audience Engagement

Tracy Cook Director of Online Marketing

ADVERTISING SALES: 508-532-1418/mmorollo@1105media.com

Matt Morollo VP/Group Publisher

Chris Kourtoglou Regional Sales Manager

William Smith National Accounts Director

Danna Vedder National Account Manager/Microsoft Account Manager

Jenny Hernandez-Asandas Director, Print Production

Serena Barnes Production Coordinator/msdnadproduction@1105media.com

1105 MEDIA

Neal Vitale President & Chief Executive Officer

Richard Vitale Senior Vice President & Chief Financial Officer

Michael J. Valenti Executive Vice President

Christopher M. Coates Vice President, Finance & Administration

Erik A. Lindgren Vice President, Information Technology & Application Development

David F. Myers Vice President, Event Operations

Jeffrey S. Klein Chairman of the Board

MSDN Magazine (ISSN 1528-4859) is published monthly by 1105 Media, Inc., 9201 Oakdale Avenue, Ste. 101, Chatsworth, CA 91311. Periodicals postage paid at Chatsworth, CA 91311-9998, and at additional mailing offices. Annual subscription rates payable in US funds are: U.S. \$35.00, International \$60.00. Annual digital subscription rates payable in U.S. funds are: U.S. \$25.00, International \$25.00. Single copies/back issues: U.S. \$10, all others \$12. Send orders with payment to: *MSDN Magazine*, P.O. Box 3167, Carol Stream, IL 60132, email MSDNmag@1105service.com or call (847) 763-9560. **POSTMASTER:** Send address changes to *MSDN Magazine*, P.O. Box 2166, Skokie, IL 60076. Canada Publications Mail Agreement No: 40612608. Return Undeliverable Canadian Addresses to Circulation Dept. or XPO Returns: P.O. Box 201, Richmond Hill, ON L4B 4R5, Canada.

Printed in the U.S.A. Reproductions in whole or part prohibited except by written permission. Mail requests to "Permissions Editor," c/o *MSDN Magazine*, 4 Venture, Suite 150, Irvine, CA 92618.

Legal Disclaimer: The information in this magazine has not undergone any formal testing by 1105 Media, Inc. and is distributed without any warranty expressed or implied. Implementation or use of any information contained herein is the reader's sole responsibility. While the information has been reviewed for accuracy, there is no guarantee that the same or similar results may be achieved in all environments. Technical inaccuracies may result from printing errors and/or new developments in the industry.

Corporate Address: 1105 Media, Inc., 9201 Oakdale Ave., Ste 101, Chatsworth, CA 91311, www.1105media.com

Media Kits: Direct your Media Kit requests to Matt Morollo, VP Publishing, 508-532-1418 (phone), 508-875-6622 (fax), mmorollo@1105media.com

Reprints: For single article reprints (in minimum quantities of 250-500), e-prints, plaques and posters contact: PARS International, Phone: 212-221-9595, E-mail: 1105reprints@parsintl.com, www.magreprints.com/QuickQuote.asp

List Rental: This publication's subscriber list, as well as other lists from 1105 Media, Inc., is available for rental. For more information, please contact our list manager, Merit Direct. Phone: 914-368-1000; E-mail: 1105media@meritdirect.com; Web: www.meritdirect.com/1105

All customer service inquiries should be sent to MSDNmag@1105service.com or call 847-763-9560.

Microsoft



Printed in the USA

LEADTOOLS[®] **MULTIMEDIA SDKs**



CAPTURE/PLAYBACK/CONVERSION/COMPRESSION

MPEG4/MPEG2/H.264/H.263/AAC/AC3 AND 100+CODECS

MKV/MXF/MP4/AVI/WMV FORMATS

MULTIPLEXERS AND DEMULTIPLEXERS

PLAY, CONVERT, AUTHOR AND BURN DVDs AND CDs

MPEG-2 TRANSPORT STREAM/DVR/KLV/RTSP

CLOUD SDK FOR DISTRIBUTED PROCESSING

IIS SMOOTH STREAMING/UDP/RTP/ENCRYPTION/WMV

DOWNLOAD OUR 60 DAY EVALUATION
WWW.LEADTOOLS.COM



SALES@LEADTOOLS.COM
800.637.1840





Visual Studio 2012: The Next Big Thing

When Microsoft launched Visual Studio 2010, I remember thinking what a truly big thing it was. Support for SharePoint, Silverlight and Windows Azure development. Infrastructure improvements like Managed Extensibility Framework and the incorporation of Windows Presentation Foundation into the Visual Studio 2010 UI. And, of course, integrated support for a major new version of the Microsoft .NET Framework.

There was enough going on in Visual Studio 2010 that some industry watchers worried it could be getting too big for its own good. And yet, here we are, a little more than two years later, contemplating a significant update to the Microsoft flagship IDE. Call it the next big thing.

There's a lot going on in Visual Studio 2012. To address it all, this month's issue of *MSDN Magazine* includes no fewer than five features focused on the new IDE.

Our lead feature this month, written by Peter Vogel, offers a hands-on tour of the big changes in Visual Studio 2012. Vogel comes away genuinely impressed. As he put it to me, Visual Studio 2012 is an IDE with appeal that reaches far beyond the nascent ranks of Windows Store app developers and even .NET devs itching to work with the .NET Framework 4.5. And you don't have to upgrade to the latest version of .NET to take advantage of the new IDE.

"I really like the consolidation and simplification changes in the UI, and think they'd make programmers more productive even without going to .NET 4.5," Vogel explains, adding, "I'm going to use the new combined Solution Explorer with its Class View features a lot. It's just a sweet design."

Vogel singled out Solution Explorer as one of the biggest surprises in Visual Studio 2012, but he has high praise for the new Page Inspector troubleshooting feature. "Page Inspector just brings so many things together in one place and makes it clear how they're driving your output. The fact that it's 'live' is very impressive, also. I can play with my CSS or my HTML and see the impact almost right away."

There's a lot going on in Visual Studio 2012. To address it all, this month's issue of *MSDN Magazine* includes no fewer than five feature articles focused on the new IDE. From an exploration of Windows Azure-focused tooling in Visual Studio 2012 to the capabilities of Microsoft Test Manager 2012, we dive into all the new features that make Visual Studio 2012 such a big thing for developers.

Ultimately, the value of Visual Studio 2012 isn't in the laundry list of new features, but rather in the way the features and capabilities of the IDE are presented to developers. And in that regard, Vogel says, Visual Studio 2012 has impressed.

"My feeling is that, for any technology you're working in, the tools you need are at hand," he says.

Welcoming a New Editorial Director

I wanted to take a moment to welcome on board Mohammad Al-Sabt, the new editorial director of *MSDN Magazine*. Al-Sabt arrived about three months after the departure of former editorial director Kit George, who left to take on an opportunity with the Bing group at Microsoft.

To say that Al-Sabt hit the ground running is a gross understatement. You see, we've been working on an extra edition of *MSDN Magazine* focused entirely on Windows 8 development. It's no small feat to produce an entire extra issue of a 100-page magazine between two regular monthly issues. But the achievement is all the greater when you consider that Al-Sabt arrived right in the middle of this critical project. He immediately jumped in with both feet and did a great job marshaling resources at Microsoft and making sure we were able to move the project forward.

The way I figure it, if Al-Sabt can get through that challenge, he's good to handle just about anything. Welcome to the magazine, Mohammad.

Visit us at msdn.microsoft.com/magazine. Questions, comments or suggestions for *MSDN Magazine*? Send them to the editor: mmeditor@microsoft.com.

© 2012 Microsoft Corporation. All rights reserved.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, you are not permitted to reproduce, store, or introduce into a retrieval system *MSDN Magazine* or any part of *MSDN Magazine*. If you have purchased or have otherwise properly acquired a copy of *MSDN Magazine* in paper format, you are permitted to physically transfer this paper copy in unmodified form. Otherwise, you are not permitted to transmit copies of *MSDN Magazine* (or any part of *MSDN Magazine*) in any form or by any means without the express written permission of Microsoft Corporation.

A listing of Microsoft Corporation trademarks can be found at microsoft.com/library/toolbar/3.0/trademarks/en-us.mspx. Other trademarks or trade names mentioned herein are the property of their respective owners.

MSDN Magazine is published by 1105 Media, Inc. 1105 Media, Inc. is an independent company not affiliated with Microsoft Corporation. Microsoft Corporation is solely responsible for the editorial contents of this magazine. The recommendations and technical guidelines in *MSDN Magazine* are based on specific environments and configurations. These recommendations or guidelines may not apply to dissimilar configurations. Microsoft Corporation does not make any representation or warranty, express or implied, with respect to any code or other information herein and disclaims any liability whatsoever for any use of such code or other information. *MSDN Magazine*, MSDN, and Microsoft logos are used by 1105 Media, Inc. under license from owner.

The Scrum project management tool your team will love to use.



Try the new **real-time visual dashboard** in **OnTime Scrum**.



OnTime Scrum

Agile project management & bug tracking

- product backlogs, releases, and sprints
- powerful user story and bug tracking
- automated burndown charts
- **real-time visual project dashboard**

**Enterprise quality software
at prices any team can afford.**

\$7 per user
per month

for teams of 11+ users

\$10 per month
for up to 10 users

special small-team pricing



OnTime Dev Suite

Also available from the **OnTime Dev Suite**:



OnTime Help Desk

Customer support for software apps



OnTime Team Wiki

Project wiki for collaborative dev teams

Visit OnTimeNow.com/MSDN for these 3 great resources:



1. Watch our popular
intro to Scrum video



2. Download and print
our free Scrum Diagram



Free Trial

3. Get started free with
OnTime Dev Suite



axosoft.com • @axosoft • 800.653.0024



Mobile Site Development, Part 4: Managing Device Profiles

In this article I'll discuss a way to classify mobile devices and build a Web site that serves different markup to different devices based on the capabilities of the device.

If you don't need to adapt the rendered markup to the capabilities of a requesting browser, building a mobile site can be a seamless experience. More often than not, though, you need to fine-tune the content you serve and adapt it to the effective capabilities of the browser. Does that sound like the reiteration of an old story? Years ago, developers faced a similar problem for desktop browsers. It was common to write Web pages that checked the type (and sometimes the version) of the browser before deciding about the markup to return. Recently, as the focus of Web programming has shifted more toward the client side, libraries such as jQuery and Modernizr have provided a significant contribution in keeping developers away from many of the browsers' differences.

It still can be difficult and expensive to build a desktop Web site that looks and works the same regardless of the browser. However, the number of desktop browsers is relatively small, and the gap between the most recent versions of all browsers is not huge. When it comes to mobile browsers, though, nearly any model of device has its own slightly different and customized browser. In addition, users may have installed a cross-device browser such as Fennec or Opera Mini. The large number of possible mobile browsers makes targeting each of them separately—as developers did with desktop browsers—a highly impractical approach. A smarter approach is to partition mobile browsers into a few classes and serve each class an ad hoc version of any given page. This approach is often referred to as multiserving.

The sample Web site for this article is built using ASP.NET MVC 3. It should be noted, though, that ASP.NET MVC 4 brings some new facilities that could make the implementation of multiserving simpler. I'll cover ASP.NET MVC 4 in relation to mobile sites in a future column.

From Fragmentation to Device Profiles

Mobile fragmentation is significant, with thousands of unique devices and hundreds of capabilities to fully describe them. You ideally need pages that can intelligently adjust to the characteristics of the requesting browsers. To achieve this, you have essentially

Figure 1 Sample Device Profiles

Device Profile	Capabilities
Smartphone	Mobile device, touch device, screen width greater than 240 pixels, based on a known OS (Android 2.1, iOS, BlackBerry 6.0 or Windows Phone).
Tablet	Mobile device and tablet device.
Mobile	Mobile device not falling into other profiles.

two possible routes. One is authoring different versions of the same page—one for each class of device it's intended to support. The other consists of having one common page template and filling it up with device-specific content on each request.

In the end, however, both approaches start from a common ground: Split your expected audience into a few categories. Then, each page of the site will provide ad hoc markup for each of the categories.

Mobile fragmentation is significant, with thousands of unique devices and hundreds of capabilities to fully describe them.

To tame device fragmentation, you should decide early on how many versions of the mobile site you intend to have. This is a key decision because it impacts the perception of the site and, ultimately, its success. It's not, therefore, a decision to take lightly, and business considerations apply—it's not simply an implementation detail or technology decision. Depending on the business case, you might decide to offer the site only to smartphones and perhaps optimize the site for, say, Windows Phone devices, in much the same way some desktop sites were showcasing the label "best viewed with XXX" a decade or so ago. More likely, though, you'll want to have at least two versions of the site—for smart and legacy devices—and maybe consider yet another version that specifically targets tablet devices or even smart TVs.

Smartphones, legacy devices and tablets are all examples of device profiles into which you split your expected audience. You don't have to write your mobile site to address thousands of devices by name; instead, you identify a few device profiles and define

Code download available at archive.msdn.microsoft.com/mag201209CuttingEdge.



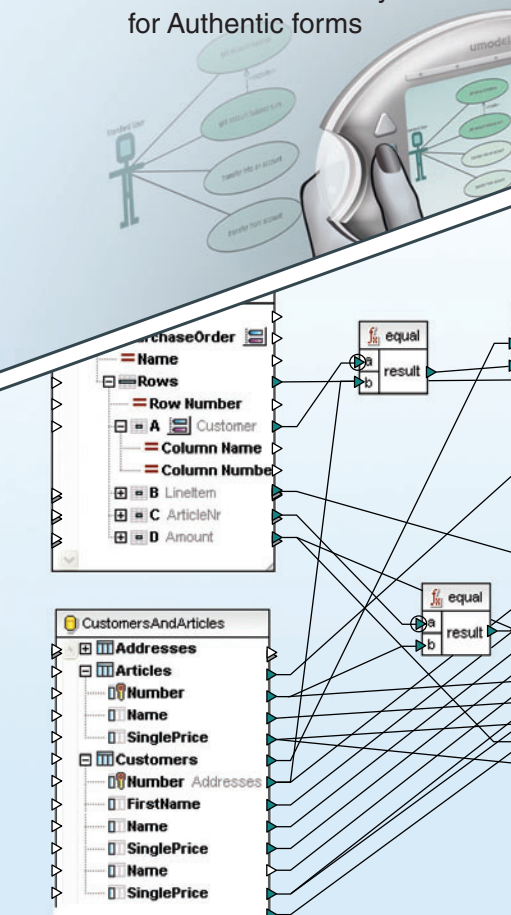
Visualize software works of art with the complete set of tools from Altova®



The MissionKit® is an integrated suite of UML, XML, and data integration tools for today's software architect.

NEW in Version 2012r2:

- Support for the EPUB e-book standard
- Sorting of data mapping results
- Code generation from UML sequence diagrams
- Support for logical files in IBM iSeries databases
- RichEdit functionality for Authentic forms



The Altova MissionKit includes multiple tools for software architects:

UModel® – UML tool for software modeling

- Support for all UML diagram types, MDA, SQL database diagrams, BPMN, SysML
- Reverse engineering and code generation in Java, C#, VB.NET

XMLSpy® – XML editor & development environment

- Support for all XML-based technologies
- Royalty-free Java, C#, C++ code generation

MapForce® – graphical data mapping tool

- Mapping between XML, databases, EDI, flat files, XBRL, Excel 2007+, Web services
- Royalty-free Java, C#, C++ code generation

Plus up to five additional tools...

 Download a 30 day free trial!

Try before you buy with a free, fully functional trial from www.altova.com



Scan to learn more about these Altova MissionKit tools.

Figure 2 A Minimal Device Profiler Implementation

```
public class DefaultDeviceProfiler : IDeviceProfiler
{
    public virtual String MobileSuffix {
        get { return "mobile"; }
    }
    public virtual Boolean IsMobile(String userAgent)
    {
        return HasAnyMobileKeywords(userAgent);
    }

    public virtual String SmartphoneSuffix {
        get { return "smartphone"; }
    }
    public virtual Boolean IsSmartphone(String userAgent)
    {
        return IsMobile(userAgent);
    }

    public virtual String TabletSuffix {
        get { return "tablet"; }
    }
    public virtual Boolean IsTablet(String userAgent)
    {
        return IsMobile(userAgent) &&
            userAgent.ContainsAny("tablet", "ipad");
    }

    public virtual Boolean IsDesktop(String userAgent)
    {
        return HasAnyDesktopKeywords(userAgent);
    }

    // Private Members
    private Boolean HasAnyMobileKeywords(String userAgent)
    {
        var ua = userAgent.ToLower();
        return (ua.Contains("midp") ||
            ua.Contains("mobile") ||
            ua.Contains("android") ||
            ua.Contains("samsung") ||
            ...
        )
    }
    private Boolean HasAnyDesktopKeywords(String userAgent)
    {
        var ua = userAgent.ToLower();
        return (ua.Contains("wow64") ||
            ua.Contains(".net clr") ||
            ua.Contains("macintosh") ||
            ...
        )
    }
}
```

which capabilities are required to join each profile. It goes without saying that there might not be fixed and universal rules that define when a device is a “smartphone” and when it’s not. There’s no ratified standard for this, and you are responsible for defining the capabilities required for a device to be classified as a smartphone in the context of your site. Also consider that the definition of a smartphone is variable by design. A Windows CE device was certainly perceived as a very smart device only five or six years ago. Today, it would be hard to include it in the smartphone category.

Project Liike—an effort of the patterns & practices group at Microsoft aimed at building a mobile reference site—splits the mobile audience into three classes, familiarly called WWW, short for Wow, Works and Whoops.

The Wow class refers to today’s rich and smart devices. The Works class refers to not-so-rich and capable devices. Finally, the Whoops class refers to any other legacy device that barely has the ability to connect to the Internet and render some basic HTML content. For more information on Project Liike, visit liike.github.com.

In this article I’ll use the following device profiles: smartphone, tablet and legacy mobile. **Figure 1** shows the (minimal) set of rules I used to accept devices in the various profiles. Note that the rules should be expanded to include more specific capabilities that depend on what your pages really need to do. For example, if you plan to use Asynchronous JavaScript and XML and HTML Document Object Model manipulation, you might want to ensure that devices have those capabilities. If you’re serving videos, you might want to ensure that devices support some given codecs. For devices that might not be able to match all of your expectations, you should provide a fallback page, and this is precisely the role of the legacy (that is, catch-all) profile.

Implementing a Simple Device Profiler

In the sample site, I formalize the content of **Figure 1** into an interface named `IDeviceProfiler`:

Figure 3 A Mobile-Aware View Engine

```
public class MobileRazorViewEngine : RazorViewEngine
{
    protected override IView CreatePartialView(
        ControllerContext context, String path)
    {
        var view = path;
        if (!String.IsNullOrEmpty(path))
            view = GetMobileViewName(context.HttpContext.Request, path);
        return base.CreatePartialView(context, view);
    }

    protected override IView CreateView(
        ControllerContext context, String path, String master)
    {
        var view = path;
        var layout = master;
        var request = context.HttpContext.Request;

        if (!String.IsNullOrEmpty(path))
            view = GetMobileViewName(request, path);
        if (!String.IsNullOrEmpty(master))
            master = GetMobileViewName(request, master);
        return base.CreateView(context, view, master);
    }

    public static String GetMobileViewName(
        HttpRequestBase request, String path)
    {
        var profiler = DependencyResolver.Current.GetService(
            typeof(IDeviceProfiler)) as IDeviceProfiler
            ?? new DefaultDeviceProfiler();

        var ua = request.UserAgent ?? String.Empty;
        var suffix = GetSuffix(ua, profiler);
        var extension = String.Format("{0}{1}",
            suffix, Path.GetExtension(path));
        return Path.ChangeExtension(path, extension);
    }

    private static String GetSuffix(String ua, IDeviceProfiler profiler)
    {
        if (profiler.IsDesktop(ua))
            return String.Empty;

        if (profiler.IsSmartphone(ua))
            return profiler.SmartphoneSuffix;
        if (profiler.IsTablet(ua))
            return profiler.TabletSuffix;
        return profiler.IsMobile(ua)
            ? profiler.MobileSuffix
            : String.Empty;
    }
}
```

LinqConnect Fits Everyone:

ORM Solution for Any Development Platform



ADO.NET and LINQ data access in new editions for Silverlight, Windows Phone and Metro



LinqConnect is a LINQ to SQL compatible ORM solution with extended functionality, support for SQL Server, Oracle, MySQL, PostgreSQL, and SQLite, its own visual model designer, seamlessly integrating to Visual Studio, and SQL monitoring tool.



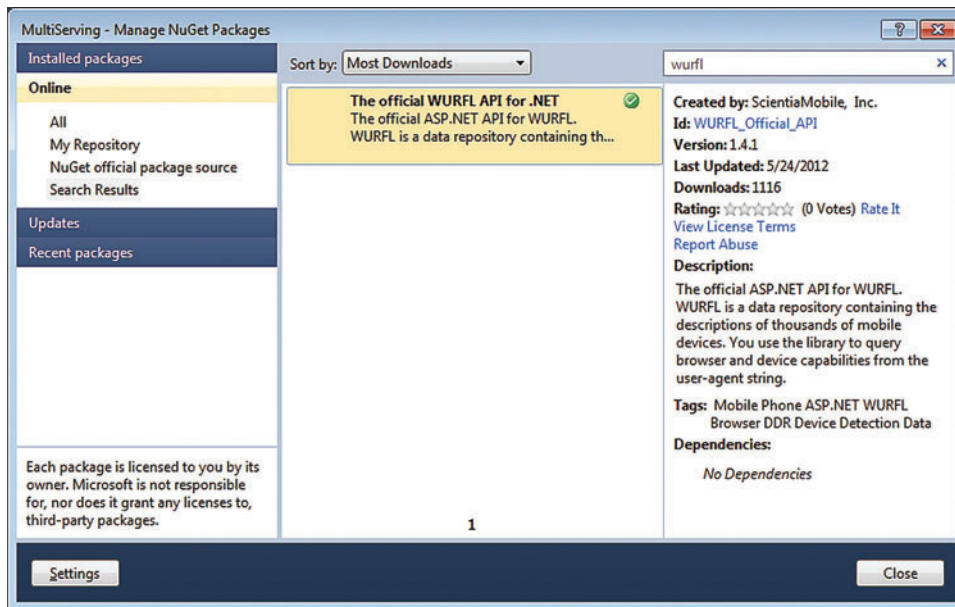


Figure 4 Adding WURFL to ASP.NET MVC via NuGet

```
public interface IDeviceProfiler
{
    Boolean IsDesktop(String userAgent);

    String MobileSuffix { get; }
    Boolean IsMobile(String userAgent);

    String SmartphoneSuffix { get; }
    Boolean IsSmartphone(String userAgent);

    String TabletSuffix { get; }
    Boolean IsTablet(String userAgent);
}
```

The suffix refers to a unique name used to differentiate views. For example, the page `index.cshtml` will be expanded to `index.smartphone.cshtml`, `index.tablet.cshtml` and `index.mobile.cshtml` for the various profiles. **Figure 2** shows a basic implementation for a device profiler object.

As you can guess from **Figure 2**, each device is identified through its user agent string. The user agent string is processed to see if it contains some keywords known to represent a mobile or desktop browser. For example, a user agent string that contains the substring “android” can be safely matched to a mobile browser. Similarly, the “wow64” substring usually refers to a desktop Windows browser. Let me say up front that while relying on user agent strings is

Figure 5 Using WURFL

```
public class MyApp : HttpApplication
{
    public static IWurflManager WurflContainer;
    protected void Application_Start()
    {
        ...
        RegisterWurfl();
        DependencyResolver.SetResolver(new SimpleDependencyResolver());
    }
    public static void RegisterWurfl()
    {
        var configurator = new ApplicationConfigurator();
        WurflContainer = WURFLManagerBuilder.Build(configurator);
    }
}
```

probably the best approach to detect device capabilities on the server side, it’s not a guarantee of success. Personally, I recently bought an Android 4.0 tablet and discovered that the embedded browser just sends out verbatim the user agent of an iPad running iOS 3.2. Device fragmentation is hard because of these issues.

Selection of View, Layout and Model

Let’s say that the device profiler can reliably tell us which profile the requesting browser belongs to. In an ASP.NET MVC site, how would you select the right view and layout from within each controller method? In any controller method that returns HTML markup, you may indicate

explicitly the name of the view and related layout. Both names can be determined in the controller method using the following code:

```
// Assume this code is from some Index method
var suffix = AnalyzeUserAgent(Request.UserAgent);
var view = String.Format("index.{0}", suffix);
var layout = String.Format("_layout.{0}", suffix);
return View(view, layout);
```

In a multiserving scenario, the differences between views for the same page may not be limited to the view template. In other words, picking up a specific pair of view and layout templates might not be enough—you might even need to pass a different view model object.

If you’re passing view model data through built-in collections such as ViewBag or ViewData, you can consider moving any code that deals with the analysis of the user agent string out of the controller. In the sample mobile site code download accompanying this article, the Index method for the home page looks like this:

```
public ActionResult Index()
{
    ViewBag.Title = "...";
    ...
    return View();
}
```

As you can see, the view is generated without an explicit indication of the name and layout. When this happens, the view engine is ultimately responsible for finalizing the view to use and its layout. The view engine is therefore a possible place to embed any logic for managing device profiles. By creating and registering a custom view engine, you isolate any logic for analyzing device profiles in a single place, and the remainder of your mobile site can be developed as a plain collection of related pages. The following code shows how to register a custom view engine in `global.asax`:

```
// Get rid of any other view engines
ViewEngines.Engines.Clear();

// Install an ad hoc mobile view engine
ViewEngines.Engines.Add(new MobileRazorViewEngine());
```

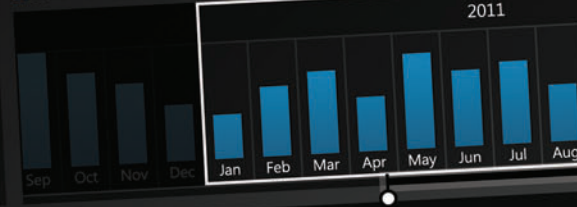
Figure 3 shows the source code of the custom (Razor-based) view engine.



CEO Dashboard January 1 - December 31, 2011



■ Sales □ Target



Total Sales



\$10,575,084

From Target: ▲ \$52.2 K (0.5%) From Prev. Period: ▼ \$92.3 K (1%)

Sales Analysis



Figure 6 A WURFL-Based Device Profiler

```
public class WurlDeviceProfiler : DefaultDeviceProfiler
{
    public override Boolean IsMobile(String ua)
    {
        var device = MyApp.WurlContainer.GetDeviceForRequest(ua);
        return device.IsWireless();
    }

    public override Boolean IsSmartphone(String ua)
    {
        var device = MyApp.WurlContainer.GetDeviceForRequest(ua);
        return device.IsWireless() && !device.IsTablet() &&
            device.IsTouch() &&
            device.Width() > 240 &&
            (device.HasOs("android", new Version(2, 1)) ||
             device.HasOs("iphone os", new Version(3, 2)) ||
             device.HasOs("windows phone os", new Version(7, 1)) ||
             device.HasOs("rim os", new Version(6, 0)));
    }

    public override Boolean IsTablet(String ua)
    {
        var device = MyApp.WurlContainer.GetDeviceForRequest(ua);
        return device.IsTablet();
    }
}
```

Before rendering a view, the view engine uses the installed device profiler to query about the profile of the requesting user agent. Based on that, the view engine switches to the most appropriate view. If the layout name is provided explicitly in the View call from within the controller, the view engine can resolve it seamlessly. If the layout name is set in `_viewStart.cshtml` (as in most ASP.NET MVC code), the view engine won't be able to resolve it because the master parameter in `CreateView` is always empty. Here's a fix to apply in `_viewStart.cshtml`:

```
@using MultiServing.ProfileManager.Mvc
@{
    const String defaultLayout = "~/Views/Shared/_Layout.cshtml";
    Layout = MobileRazorViewEngine.GetMobileViewName(
        Context.Request, defaultLayout);
}
```

What if you use strongly typed views, and the various mobile views for the same page (smartphone, tablet and so on) each requires its own view model? In this case, you might want to build a worker component that analyzes the user agent and returns the view/layout name,



Figure 7 Tablets, Smartphones and Plain Mobile Devices Accessing the Sample Site

and use this component from within each controller method. As I see things, if you need to parse the user agent right at the controller level to decide about the view model, then relying on a custom view engine is redundant because you already know which view to call.

Beyond this point, it's all about using an appropriate device profiler and building multiple HTML page templates.

Configuring WURFL in ASP.NET MVC

As mentioned in previous installments of this series, the Wireless Universal Resource File (WURFL) is a popular Device Description Repository (DDR) used in the back ends of Google and Facebook mobile sites. WURFL offers a multiplatform API and can be easily plugged into any ASP.NET MVC project using NuGet (see **Figure 4**).

WURFL adds an XML database to your project that contains device information. The database should be loaded into memory at application startup and provides nearly instant access to device profiles. In `global.asax`, you add the code shown in **Figure 5**.

In **Figure 6**, you see an `IDeviceProfiler` component that uses WURFL to detect smartphones and tablets. You resolve the profiler via a custom dependency resolver. (See the accompanying source code for details about the resolver.)

WURFL adds an XML database to your project that contains device information.

The method `GetDeviceForRequest` queries the WURFL database and returns an `IDevice` object that can be queried using a relatively fluent syntax. Note that methods such as `IsTouch`, `IsTablet` and `HasOs` are actually extension methods built over the native WURFL API that you find in the sample project. As an example, here's the code for `IsTablet`:

```
public static Boolean IsTablet(this IDevice device)
{
    return device.GetCapability("is_tablet").ToBool();
}
```

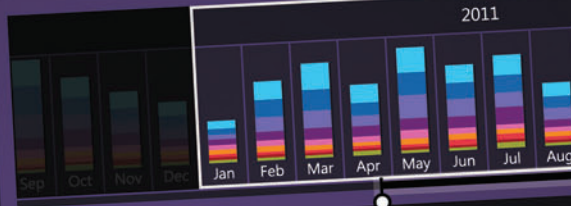
I've discussed in this column a concrete example of an ASP.NET MVC mobile site built to provide a different experience on a variety of devices: smartphones, tablets and legacy mobile devices, as shown in **Figure 7**. I suggest you download the source code and run it in Internet Explorer (or other browsers), switching to different user agents. You can also test the site live at www.expoware.org/amse/ddr. Note that accessing the site from a desktop browser results in the message: "This site is not available on desktop browsers. Try using a mobile or tablet browser."

DINO ESPOSITO is the author of "Architecting Mobile Solutions for the Enterprise" (Microsoft Press, 2012) and "Programming ASP.NET MVC 3" (Microsoft Press, 2011), and coauthor of "Microsoft .NET: Architecting Applications for the Enterprise" (Microsoft Press, 2008). Based in Italy, Esposito is a frequent speaker at industry events worldwide. Follow him on Twitter at twitter.com/despos.

THANKS to the following technical experts for reviewing this article:
Erik Porter and Pranav Rastogi



Expense Dashboard January 1 - December 31, 2011



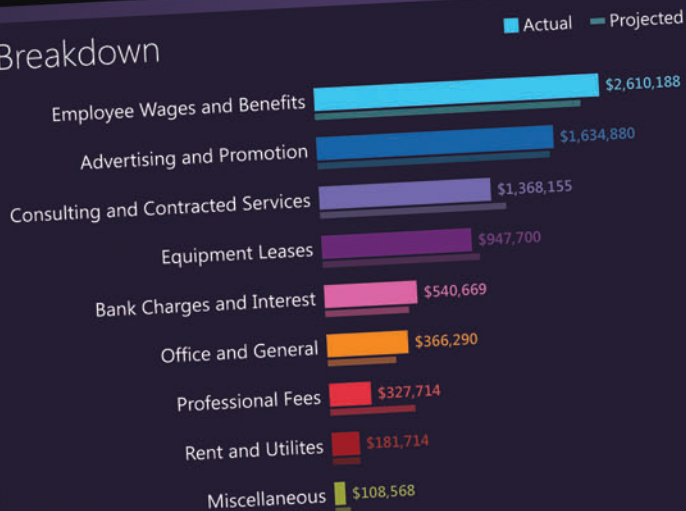
Total Expenses

\$9,221,481

From Target

▲ \$6,143,435
(5%)

Breakdown



Sales Analysis





The Pursuit of Efficient and Composable Asynchronous Systems

The implementation of computer hardware heavily influenced the design of the C programming language to follow an imperative approach to computer programming. This approach describes a program as a sequence of statements that embody the program's state. This was an intentional choice by C designer Dennis Ritchie. It allowed him to produce a viable alternative to assembly language. Ritchie also adopted a structured and procedural design, which has proven to be effective at improving the quality and maintainability of programs, leading to the creation of vastly more sophisticated and powerful system software.

A particular computer's assembly language typically consists of the set of instructions supported by the processor. The programmer can refer to registers—literally small amounts of memory on the processor itself—as well as addresses in main memory. Assembly language will also contain some instructions for jumping to different locations in the program, providing a simplistic way to create reusable routines. In order to implement functions in C, a small amount of memory called the “stack” is reserved. For the most part, this stack, or call stack, stores information about each function that's called so the program can automatically store state—both local and shared with its caller—and know where execution should resume once the function completes. This is such a fundamental part of computing today that most programmers don't give it a second thought, yet it's an incredibly important part of what makes it possible to write efficient and comprehensible programs. Consider the following code:

```
int sum(int a, int b) { return a + b; }

int main()
{
    int x = sum(3, 4);
    return sum(x, 5);
}
```

Given the assumption of sequential execution, it's obvious—if not explicit—what the state of the program will be at any given point. These functions would be meaningless without first assuming there's some automatic storage for function arguments and return values, as well as some way for the program to know where to resume execution when the function calls return. For C and C++ programmers, it's the stack that makes this possible and allows us to write simple and efficient code. Unfortunately, it's also our dependency on the stack that causes C and C++ programmers a world of hurt when it comes to asynchronous programming. Traditional systems programming languages such as C and C++ must adapt in order to remain competitive and productive in a world filled

with increasingly asynchronous operations. Although I suspect C programmers will continue to rely on traditional techniques to accomplish concurrency for some time, I'm hopeful that C++ will evolve more quickly and provide a richer language with which to write efficient and composable asynchronous systems.

Traditional systems programming languages such as C and C++ must adapt in order to remain competitive and productive in a world filled with increasingly asynchronous operations.

Last month I explored a simple technique that you can use today with any C or C++ compiler to implement lightweight cooperative multitasking by simulating coroutines with macros. Although adequate for the C programmer, it presents some challenges for the C++ programmer, who naturally and rightly relies on local variables among other constructs that break the abstraction. In this column, I'm going to explore one possible future direction for C++ to directly support asynchronous programming in a more natural and composable way.

Tasks and Stack Ripping

As I mentioned in my last column (msdn.microsoft.com/magazine/jj553509), concurrency doesn't imply threaded programming. This is a conflation of two separate issues but is prevalent enough to cause some confusion. Because the C++ language originally didn't provide any explicit support for concurrency, programmers naturally used different techniques to achieve the same. As programs became more complex, it became necessary—and perhaps obvious—to divide programs into logical tasks. Each task would be a sort of mini program with its own stack. Typically, an OS implements this with threads, and each thread is given its own stack. This allows tasks to run independently and often preemptively depending on the

Powerful Tools for Developers

v4.5!



High-Performance PDF Printer Driver



- Create accurate PDF documents in a fraction of the time needed with other tools
- WHQL tested for all Windows 32 and 64-bit platforms
- Produce fully compliant PDF/A documents
- Standard PDF features included with a number of unique features
- Interface with any .NET or ActiveX programming language

v4.5!



PDF Editor for .NET, now Webform Enabled

- Edit, process and print PDF 1.7 documents programmatically
- Fast and lightweight 32 and 64-bit managed code assemblies for Windows, WPF and Web applications
- Support for dynamic objects such as edit-fields and sticky-notes
- Save image files directly to PDF, with optional OCR
- Multiple image compression formats such as PNG, JBIG2 and TIFF

New!

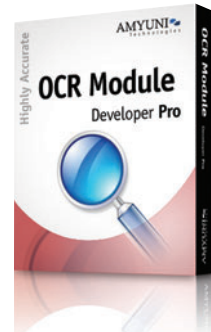


PDF Integration into Silverlight Applications

- Server-side PDF component based on the robust Amyuni PDF Creator ActiveX or .NET components
- Client-side C# Silverlight 3 control provided with source-code
- Optimization of PDF documents prior to converting them into XAML
- Conversion of PDF edit-boxes into Silverlight TextBox objects
- Support for other document formats such as TIFF and XPS



OCR Module available with royalty-free licensing!



The new OCR Module from Amyuni enables developers to:

- Convert non-searchable PDF files into searchable PDFs
- Create searchable PDF documents out of various image formats such as multi-page TIFF, JPEG or PNG while applying text recognition
- Compress image based PDF documents using high compression JBIG2 or more standard CCITT, JPEG and PNG compression formats

The Amyuni OCR module is based on the Tesseract Library with the Amyuni PDF technology being used to process and create the PDF documents.

Learn more at www.amyuni.com

More Development Tools Available at:

www.amyuni.com

USA and Canada
Toll Free: 1 866 926 9864
Support: (514) 868 9227
Info: sales@amyuni.com

Europe
Sales: (+33) 1 30 61 07 97
Support: (+33) 1 30 61 07 98
Customizations: management@amyuni.com

AMYUNI Technologies

All trademarks are property of their respective owners. © 1999-2010 AMYUNI Technologies. All rights reserved.

scheduling policy and the availability of multiple processing cores. However, each task, or mini C++ program, is simple to write and can execute sequentially thanks to its stack isolation and the state the stack embodies. This one-thread-per-task approach has some obvious limitations, however. The per-thread overhead is prohibitive in many cases. Even if it were not so, the lack of cooperation between threads leads to much complexity due to the necessity to synchronize access to shared state or communicate between threads.

Although C++11 now has much to say about concurrency in the standard library, it's still largely silent in the language itself.

Another approach that has gained much popularity is event-driven programming. It's perhaps more evident that concurrency doesn't imply threaded programming when you consider the many examples of event-driven programming in UI development and libraries relying on callback functions to implement a form of cooperative task management. But the limitations of this approach are at least as problematic as those for the one-thread-per-task approach. Immediately your clean, sequential program becomes a web—or, optimistically, a spaghetti stack—of callback functions instead of a cohesive sequence of statements and function calls. This is sometimes called *stack ripping*, because a routine that was previously a single function call is now ripped into two or more functions. This in turn also frequently leads to a ripple effect throughout a program. Stack ripping is disastrous if you care at all about complexity. Instead of one function, you now have at least two. Instead of relying on automatic storage of local variables on the stack, you must now explicitly manage the storage for this state, as it must survive between one stack location and another. Simple language constructs such as loops must be rewritten to accommodate this separation. Finally, debugging stack-ripped programs is much harder because the state of the program is no longer embodied in the stack and must often be manually “reassembled” in the programmer's head. Consider the example of a simple flash storage driver for an embedded system from my last column, expressed with synchronous operations to provide obviously sequential execution:

```
void storage_read(void * buffer, uint32 size, uint32 offset);
void storage_write(void * buffer, uint32 size, uint32 offset);

int main()
{
    uint8 buffer[1024];
    storage_read(buffer, sizeof(buffer), 0);
    storage_write(buffer, sizeof(buffer), 1024);
}
```

It's not hard to figure out what's going on here. A 1KB buffer that's backed by the stack is passed to the `storage_read` function, suspending the program until the data has been read into the buffer. This same buffer is then passed to the `storage_write` function, suspending the program until the transfer completes. At this

point, the program returns safely, automatically reclaiming the stack space that was used for the copy operation. The obvious downside is that the program isn't doing useful work while suspended, waiting for the I/O to complete.

In my last column I demonstrated a simple technique for implementing cooperative multitasking in C++ in a way that lets you return to a sequential style of programming. However, without the ability to use local variables, it's somewhat limited. Although stack management remains automatic as far as function calls and returns go, the loss of automatic stack variables is a pretty severe limitation. Still, it beats full-blown stack ripping. Consider what the preceding code might look like using a traditional event-driven approach and you can plainly see stack ripping in action. First, the storage functions would need to be redeclared to accommodate some sort of event notification, commonly by means of a callback function:

```
typedef void (* storage_done)(void * context);
```

```
void storage_read(void * b, uint32 s, uint32 o, storage_done, void * context);
void storage_write(void * b, uint32 s, uint32 o, storage_done, void * context);
```

Next, the program itself would need to be rewritten to implement the appropriate event handlers:

```
void write_done(void *)
{
    ... signal completion ...
}

void read_done(void * b)
{
    storage_write(b, 1024, 1024, write_done, nullptr);
}

int main()
{
    uint8 buffer[1024];

    storage_read(buffer, sizeof(buffer), 0, read_done, buffer);

    ... wait for completion signal ...
}
```

C++11 has made some notable steps toward an elegant solution, but we're not quite there yet.

This is clearly far more complex than the earlier synchronous approach, yet it's very much the norm today among C and C++ programs. Notice how the copy operation that was originally confined to the main function is now spread over three functions. Not only that, but you almost need to reason about the program in reverse, as the `write_done` callback needs to be declared before `read_done` and it needs to be declared before the main function. Still, this program is somewhat simplistic, and you should appreciate how this would only get more cumbersome as the “chain of events” was fully realized in any real-world application.

C++11 has made some notable steps toward an elegant solution, but we're not quite there yet. Although C++11 now has much to say about concurrency in the standard library, it's still largely silent in the language itself. The libraries themselves also don't go far enough to allow the programmer to write more complex composable



The 1st toolset enabling you to be successful
in the marketplace or enterprise

RadControls for Metro

Build for Windows 8 with either XAML or HTML



www.telerik.com/win8

telerik
deliver more than expected

and asynchronous programs easily. Nevertheless, great work has been done, and C++11 provides a good foundation for further refinements. First, I'm going to show you what C++11 offers, then what's missing and, finally, a possible solution.

Closures and Lambda Expressions

In general terms, a closure is a function coupled with some state identifying any nonlocal information that the function needs in order to execute. Consider the TrySubmitThreadPoolCallback function I covered in my thread pool series last year (msdn.microsoft.com/magazine/hh335066):

```
void CALLBACK callback(PTP_CALLBACK_INSTANCE, void * state) { ... }

int main()
{
    void * state = ...
    TrySubmitThreadPoolCallback(callback, state, nullptr);
    ...
}
```

Closures as a first-class concept rose to fame in the functional programming world, but C++11 has made strides to support the concept as well, in the form of lambda expressions.

Notice how the Windows function accepts both a function as well as some state. This is in fact a closure in disguise; it certainly doesn't look like your typical closure, but the functionality is the same. Arguably, function objects achieve the same end. Closures as a first-class concept rose to fame in the functional programming world, but C++11 has made strides to support the concept as well, in the form of lambda expressions:

```
void submit(function<void()> f) { f(); }

int main()
{
    int state = 123;
    submit([state]() { printf("%d\n", state); });
}
```

In this example there's a simple submit function that we can pretend will cause the provided function object to execute in some other context. The function object is created from a lambda expression in the main function. This simple lambda expression includes the necessary attributes to qualify as a closure and the conciseness to be convincing. The [state] part indicates what state is to be "captured," and the rest is effectively an anonymous function that has access to this state. You can plainly see that the compiler will create the moral equivalent of a function object to pull this off. Had the submit function been a template, the compiler might even have optimized away the function object itself, leading to performance gains in addition to the syntactic gains. The bigger question, however, is whether this is really a valid closure. Does the lambda

expression really close the expression by binding the nonlocal variable? This example should clarify at least part of the puzzle:

```
int main()
{
    int state = 123;
    auto f = [state]() { printf("%d\n", state); };
    state = 0;
    submit(f);
}
```

This program prints "123" and not "0" because the state variable was captured by value rather than by reference. I can, of course, tell it to capture the variable by reference:

```
int main()
{
    int state = 123;
    auto f = [&]() { printf("%d\n", state); };
    state = 0;
    submit(f);
}
```

Here I'm specifying the default capture mode to capture variables by reference and letting the compiler figure out that I'm referring to the state variable. As is expected, the program now dutifully prints "0" rather than "123." The problem, of course, is that the storage for the variable is still bound to the stack frame in which it was declared. If the submit function delays execution and the stack unwinds, then the state would be lost and your program would be incorrect.

Dynamic languages such as JavaScript get around this problem by merging the imperative world of C with a functional style that relies far less on the stack, with each object essentially being an unordered associative container. C++11 provides the shared_ptr and make_shared templates, which provide efficient alternatives even if they're not quite as concise. So, lambda expressions and smart pointers solve part of the problem by allowing closures to be defined in context and allowing state to be freed from the stack without too much syntactic overhead. It's not ideal, but it's a start.

Promises and Futures

At first glance, another C++11 feature called futures might appear to provide the answer. You can think of futures as enabling explicitly asynchronous function calls. Of course, the challenge is in defining what exactly that means and how it gets implemented. It's easier to explain futures with an example. A future-enabled version of the original synchronous storage_read function might look like this:

```
// void storage_read(void * b, uint32 s, uint32 o);
future<void> storage_read(void * b, uint32 s, uint32 o);
```

Notice that the only difference is that the return type is wrapped in a future template. The idea is that the new storage_read function will begin or queue the transmission before returning a future object. This future can then be used as a synchronization object to wait for the operation to complete:

```
int main()
{
    uint8 buffer[1024];

    auto f = storage_read(buffer, sizeof(buffer), 0);
    ...
    f.wait();
    ...
}
```

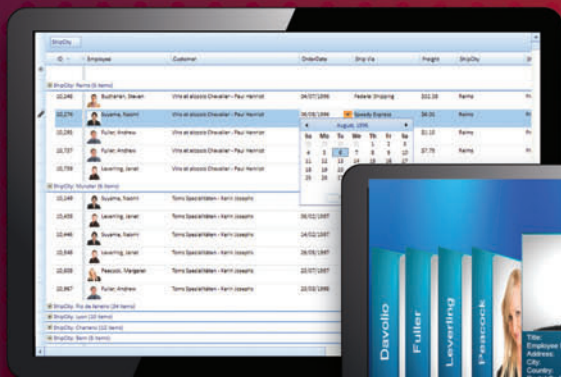
This might be called the consumer end of the asynchronous equation. The storage_read function abstracts away the provider end and is equally simple. The storage_read function would need to create a promise and queue it along with the parameters of the

5 YEARS OF EXCELLENCE



XCEED
DataGrid
for WPF

Mature, feature-packed, and lightning-fast. The most adopted and trusted WPF datagrid.



IBM®
U2 SystemBuilder™

"IBM U2 researched existing third-party solutions and identified Xceed DataGrid for WPF as the most suitable tool. The datagrid's performance and solid foundation take true advantage of WPF and provide the extensibility required for IBM U2 customers to take their applications to the next level in presentation design and styling."

Vincent Smith
U2 Tools Product Manager at IBM

Microsoft®
Visual Studio® Team System 2010

"Using Xceed DataGrid for WPF in Microsoft Visual Studio System 2010 helped us greatly reduce the time and resources necessary for developing all the data presentation feature we needed. Working with Xceed has been a pleasure."

Norman Guadagno
Director of Product Marketing
for Microsoft Visual Studio Team System



Theme your entire app in minutes. Flawless styles for all official WPF controls.



Incredible streaming technology. Speed up your app and say goodbye to paging.



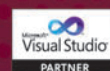
The world's first streaming listbox. Simple, drop-in upgrade to the WPF listbox.



Fast and fluid, with ground-breaking streaming technology.

NEW

NEW



request and return the associated future. Again, this is easier to understand in code:

```
future<void> storage_read(void * b, uint32 s, uint32 o)
{
    promise<void> p;
    auto f = p.get_future();
    begin_transfer(move(p), b, s, o);
    return f;
}
```

Once the operation completes, the storage driver can signal to the future that it's ready:

```
p.set_value();
```

What value is this? Well, no value at all, because we're using the promise and future specializations for void, but you can imagine a file system abstraction built on top of this storage driver that might include a `file_read` function. This function might need to be called without knowing the size of a particular file. It could then return the actual number of bytes transferred:

```
future<int> file_read(void * b, uint32 s, uint32 o);
```

In this scenario, a promise with type `int` would also be used, thus providing a channel through which to communicate the number of bytes actually transferred:

```
promise<int> p;
auto f = p.get_future();
...
p.set_value(123);
...
f.wait();
printf("bytes %d\n", f.get());
```

The problem with futures and promises is that they don't go far enough and arguably are completely flawed.

The future provides the `get` method through which the result may be obtained. Great, we have a way of waiting on the future, and all our problems are solved! Well, not so fast. Does this really solve our problem? Can we kick off multiple operations concurrently? Yes. Can we easily compose aggregate operations or even just wait on any or all outstanding operations? No. In the original synchronous example, the read operation necessarily completed before the write operation began. So futures do not in fact get us very far. The problem is that the act of waiting on a future is still a synchronous operation and there's no standard way to compose a chain of events. There's also no way to create an aggregate of futures. You might want to wait for not one but any number of futures. You might need to wait for all futures or just the first one that's ready.

Futures in the Future

The problem with futures and promises is that they don't go far enough and arguably are completely flawed. Methods such as `wait` and `get`, both of which block until the result is ready, are antithetical to concurrency and asynchronous programming. Instead of `get` we need something such as `try_get` that will attempt to retrieve the result if it's available, but return immediately, regardless:

```
int bytes;

if (f.try_get(bytes))
{
    printf("bytes %d\n", bytes);
}
```

Because futures are always returned, you might prefer a more implicit but equivalent style.

Going further, futures should provide a continuation mechanism so we can simply associate a lambda expression with the completion of the asynchronous operation. This is when we start to see the composability of futures:

```
int main()
{
    uint8 buffer[1024];

    auto fr = storage_read(buffer, sizeof(buffer), 0);

    auto fw = fr.then([&]()
    {
        return storage_write(buffer, sizeof(buffer), 1024);
    });

    ...
}
```

The `storage_read` function returns the read future (`fr`), and a lambda expression is used to construct a continuation of this future using its `then` method, resulting in a write future (`fw`). Because futures are always returned, you might prefer a more implicit but equivalent style:

```
auto f = storage_read(buffer, sizeof(buffer), 0).then([&]()
{
    return storage_write(buffer, sizeof(buffer), 1024);
});
```

In this case there's only a single explicit future representing the culmination of all the operations. This might be called sequential composition, but parallel AND and OR composition would also be essential for most nontrivial systems (think `WaitForMultipleObjects`). In this case we would need a pair of `wait_any` and `wait_all` variadic functions. Again, these would return futures, allowing us to provide a lambda expression as a continuation of the aggregate using the `then` method as before. It might also be useful to pass the completed future to the continuation in cases where the specific future that completed isn't apparent.

For a more exhaustive look at the future of futures, including the essential topic of cancellation, please look at Artur Laksberg and Niklas Gustafsson's paper, "A Standard Programmatic Interface for Asynchronous Operations," at bit.ly/MEgzhn.

Stay tuned for the next installment, where I'll dig deeper into the future of futures and show you an even more fluid approach to writing efficient and composable asynchronous systems. ■

KENNY KERR is a software craftsman with a passion for native Windows development. Reach him at kennykerr.ca.

THANKS to the following technical expert for reviewing this article:
Artur Laksberg



Dazzling Design. Developer Delivered.

Beautiful design and rich user experiences are moving to the center of the development conversation. Your existing applications now must run on an array of mobile devices, like iPads, slates and mobile phones. Building applications designed for the device they'll run on requires more focus than ever on great design principles. The design time visualization, dynamic themes and Touch-enabled tools in **DXperience 12.1** by DevExpress help you realize your vision.

Your users are ready.
Ensure you're ready, too.

Download your
free 30-day trial at
DevExpress.com

DXv2

The next generation of inspiring tools. Today.





Moving Existing Projects to EF 5

Among changes to the Microsoft .NET Framework 4.5 are a number of modifications and improvements to the core Entity Framework APIs. Most notable is the new way in which Entity Framework caches your LINQ to Entities queries automatically, removing the performance cost of translating a query into SQL when it's used repeatedly. This feature is referred to as Auto-Compiled Queries, and you can read more about it and other performance improvements in the team's blog post, "Sneak Preview: Entity Framework 5 Performance Improvements," at bit.ly/zlx21L. A bonus of this feature is that it's controlled by the Entity Framework API within the .NET Framework 4.5, so even .NET 4 applications using Entity Framework will benefit "for free" when run on machines with .NET 4.5 installed.

Other useful new features built into the core API require some coding on your part, including support for enums, spatial data types and table-valued functions. The Visual Studio 2012 Entity Data Model (EDM) designer has some new features as well, including the ability to create different views of the model.

I do most of my EF-related coding these days using the DbContext API, which is provided, along with Code First features, separately from the .NET Framework. These features are Microsoft's way of more fluidly and frequently enhancing Entity Framework, and they're contained in a single library named EntityFramework.dll, which you can install into your projects via NuGet.

To take advantage of enum support and other features added to EF in the .NET Framework 4.5, you'll need the compatible version of EntityFramework.dll, EF 5. The first release of this package has the version number 5.

I have lots of applications that use EF 4.3.1. This version includes the migration support introduced in EF 4.3, plus a few minor tweaks that were added shortly after. In this column I'll show you how to move an application that's using EF 4.3.1 to EF 5 to take advantage of the new enum support in .NET 4.5. These steps also apply to projects that are using EF 4.1, 4.2 or 4.3.

I'll start with a simple demo-ware solution that has a project for the DomainClasses, another for the DataLayer and one that's a console application, as shown in **Figure 1**.

This solution was built in Visual Studio 2010 using the .NET Framework 4 and the EF 4.3.1 version of EntityFramework.dll.

This article uses the Visual Studio 2012 release candidate for screenshots.

Code download available at archive.msdn.microsoft.com/mag201209DataPoints.

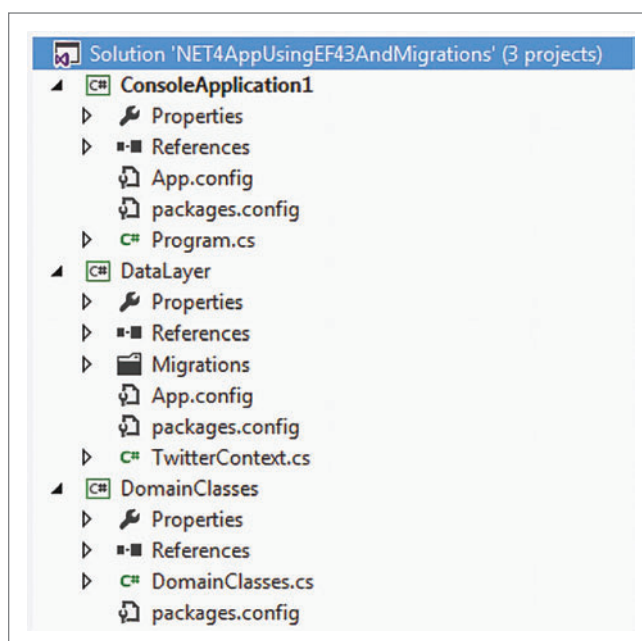


Figure 1 The Existing Solution That Uses EF 4.3.1

The DomainClasses project has two classes stuffed into a single file, shown in **Figure 2**, using a popular theme for sample code: Twitter. The classes are Tweeter and Tweet.

This project uses Data Annotations not only to add validations (such as RegularExpression) but also to define some of the configuration—MaxLength, MinLength and Column. The last, Column, specifies the column name in the database table to which the fields Experience and Rating map.

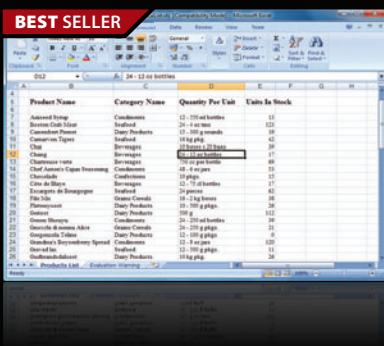
All three projects reference EntityFramework.dll (version 4.3.1). Typically, I keep the EntityFramework.dll and any database knowledge out of my domain classes, but I've chosen to include it in this example for demonstrative purposes. The MaxLength, MinLength and Column attributes are in the same namespace as the validations (System.ComponentModel.DataAnnotations), but they're part of the EntityFramework assembly.

Also notable in the domain classes is the fact that I have two properties that beg to use enums: Tweeter.Experience, which leans on a string for its value, and Tweet.Rating, which uses a numeric value. It's up to the developer coding against these classes to ensure that the users have the proper values available to them. Why no enums? Because the core Entity Framework API in the

**GdPicture.NET** from \$3,919.47

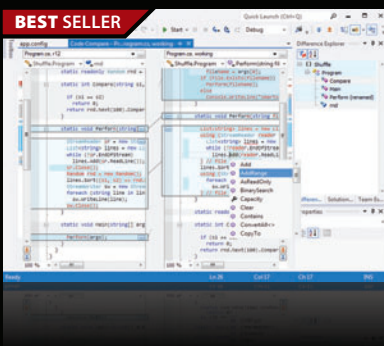
A full-featured document-imaging and image processing toolkit for software developers.

- Scan (TWAIN & WIA), process, create, view, edit, annotate, OCR images & PDF files and print documents within your Windows & Web applications
- Read, write and convert vector & raster images in more than 90 formats, including PDF
- Add forms processing, 1D and 2D Barcode recognition features to your programs
- GdPicture SDKs are AnyCPU, thread-safe and 100% royalty-free

**Aspose.Total for .NET** from \$2,449.02

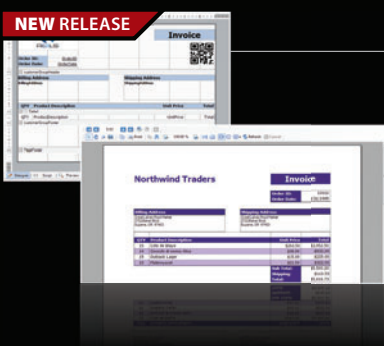
Every Aspose .NET component in one package.

- Programmatically manage popular file formats including Word, Excel, PowerPoint and PDF
- Add charting, email, spell checking, barcode creation, OCR, diagramming, imaging, project management and file format management to your .NET applications
- Common uses also include mail merge, adding barcodes to documents, building dynamic Excel reports on the fly and extracting text from PDF files

**Code Compare Pro** from \$48.95

An advanced visual file comparison tool with Visual Studio integration.

- Code oriented comparison, including syntax highlighting, unique structure and lexical comparison algorithms, for the most popular programming languages
- Smooth Visual Studio integration to develop and merge within one environment in the context of current solution, using native IDE editors
- Three-way file merge, folder comparison and synchronization

**ActiveReports 7** from \$783.02

The fast and flexible reporting engine has gotten even better.

- New page layout designer - offers precise design tools for the most complex form reports such as tax forms, insurance forms and investment forms, etc.
- More controls - updated Barcode, Matrix, Calendar and Table controls plus data visualization features
- More customization options - redesigned viewer and designer controls

Figure 2 The Original Domain Classes

```
using System.ComponentModel.DataAnnotations;
namespace DataPointsDemo.DomainClasses
{
    public class Tweeter
    {
        public Tweeter()
        {
            Tweets = new List<Tweet>();
        }
        public int Id { get; set; }
        [Required]
        public string Name { get; set; }
        [MaxLength(10), Column("ExperienceCode")]
        public string Experience { get; set; }
        [MaxLength(30), MinLength(5)]
        public string UserName { get; set; }
        [RegularExpression(@"(\\w[-_\\w]*\\w[-_\\w]*\\w\\.\\w{2,3})")]
        public string Email { get; set; }
        public string Bio { get; set; }
        public DateTime CreateDate { get; set; }
        public byte[] Avatar { get; set; }
        public ICollection<Tweet> Tweets { get; set; }
        public string AliasPlusName
        { get { return Name + "(" + UserName + ")"; } }
    }

    public class Tweet
    {
        public int Id { get; set; }
        public DateTime CreateDate { get; set; }
        public string Content { get; set; }
        [Range(1, 5), Column("RatingCode")]
        public int Rating { get; set; }
        public Tweeter Alias { get; set; }
        public int AliasId { get; set; }
    }
}
```

.NET Framework 4 doesn't support enums. But because this was the most-requested feature for Entity Framework and is now part of the .NET Framework 4.5 (and supported by Code First in EF 5), I can use it. So let's update the solution.

Although I've opened my solution in Visual Studio 2012 RC, it's still targeting .NET 4. The first thing I must do is target my three projects to .NET 4.5, which I can do in the Properties window of each project (see **Figure 3**). You have to do this one at a time, so if you have a lot of projects you might want to use a script to run against the project files directly.

It's important to do this step before updating to EF 5. I learned this the hard way and will explain shortly why this is.

Once the projects are targeting the .NET Framework 4.5, you can upgrade to EF 5. Because multiple projects use this assembly, you'll want to manage the packages for the entire solution rather than updating one project at a time. Manage NuGet Packages is available from the solution's context menu in Solution Explorer. This will open up the package manager UI. On the left, select Updates. In the middle pane, if you have a current version of the package manager, you'll see a dropdown box with the options Stable Only and Include Prerelease. If, like me, you're doing this prior to the full release of .NET 4.5 and EF 5, you'll need to select Include Prerelease. In my particular solution, EntityFramework is the only package that needs updating so that's what's showing up, as you can see in **Figure 4**. If you're a fan of working in the package manager console, you can type in "Install-Package EntityFramework -prerelease," but you'd have to do this separately for each project.

From the wizard, once you trigger the package update, NuGet will ask you which projects to update. Even though all three of my projects use Entity Framework 4.3.1, I'm only going to update the ConsoleApplication and DataLayer, so I'll deselect DomainClasses. You can watch the status box as it tells you what steps it's taking. When the update is complete, just close the package manager.

One Package, Two DLLs

Updating to EF 5 affected the two projects in a few ways. First, it replaced the 4.3.1 version of EntityFramework.dll with 5. You should verify this in all of the projects you update. This demonstrates why it's important to switch to the .NET Framework 4.5 prior to executing the package-update step. The EF 5 package contains two DLLs. One is version 5, which contains all of the DbContext API and Code First features and is compatible with .NET 4.5. The other file is version 4.4. This is the one that remains compatible with .NET 4. By including this DLL in the package, the team avoided the need to maintain two separate NuGet packages for you to worry about. After EF 5 releases, you'll install the same EF 5 package whenever you want DbContext or Code First support. The package will take care of making sure you have the correct version installed in your project—whether the project is .NET 4 or .NET 4.5.

The first time I did this update, I hadn't upgraded my projects to .NET 4.5 before the EF update. I couldn't get the new features to work and was very confused. Then I noticed that the version of EntityFramework.dll was 4.4, which made me more confused. Eventually, I browsed to the package files in the solution and saw that I had two packages, and understood my mistake.

The EF 5 update also modified the app.config file in the Console project and created an app.config file in the DataLayer project. Because my original solution let Code First use its default behavior for automatically detecting the relevant database, I had no connection string or connection factory information in the config file. EF 5 installation added the following section into the <entityFramework> section of the file:

```
<defaultConnectionFactory
  type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework">
  <parameters>
    <parameter
      value="Data Source=(localdb)\v11; Integrated Security=True;
      MultipleActiveResultSets=True" />
    </parameters>
  </defaultConnectionFactory>
```

Additionally, it updated app.config's reference to the EF assembly to reflect the new version number.

Projects that have no config file will get a new app.config with the EF 5 default configuration. That's why the DataLayer project has an app.config after the update. But I don't need a config file in that project, so I'll just delete that file.

What About the DomainClasses Project?

When updating, I skipped the project with the domain classes. The only reason I needed the EntityFramework.dll in the previous version of my solution was to have access to the Data Annotations that were specific to EF. Those have now been moved to the .NET Framework 4.5 assembly, System.ComponentModel.DataAnnotations.dll, to join the other data annotations. So I no longer

ONLY **\$20/mo** with 30 DAY FREE TRIAL!



IMPROVE THE SPEED OF YOUR SOFTWARE PROJECTS TODAY WITH TFS PROXY SERVER!

DiscountASP.NET provides TFS hosting to development teams in over 75 countries, and we understand that software development should not be slowed by geographic limitations. But the fact is, geographical latency does impact development teams that are dispersed throughout the world. The DiscountASP.NET TFS Proxy Servers diminish latency issues by saving local copies of source control files to caching proxy servers that are closer to your team members. So, for example, if you are a North American firm with developers located in Europe, our TFS proxy servers located in London will help reduce latency and increase efficiency. With DiscountASP.NET TFS Proxy Service, all of your team members, no matter where they are, can enjoy the speedier workflow that a server near their location provides.

www.DiscountASP.NET/TFS/MSDN

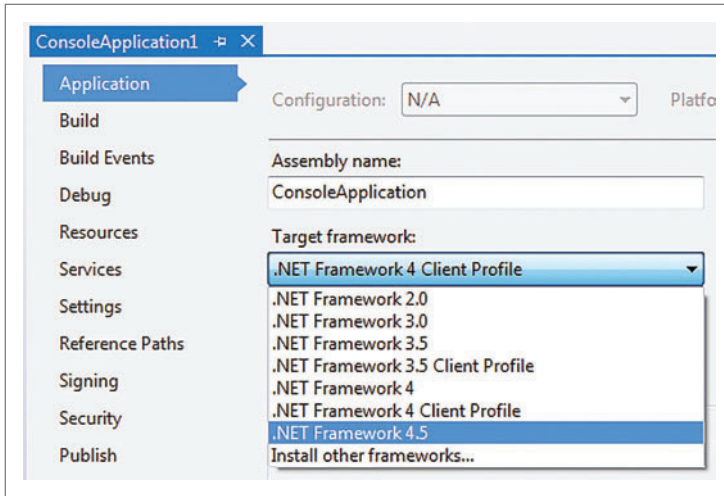


Figure 3 Changing a .NET Framework 4 Project to .NET Framework 4.5

need to reference EF from that project. In fact, I can now uninstall the EntityFramework reference from that project. Rather than using the package manager UI, I prefer to open up the package manager console window, ensure that I'm targeting the Domain-Classes project and then type "uninstall-package entityframework" to remove the package from that project.

You can specify the enum to be a different type and Code First will honor that.

There's one more step, however. Opening the file with the classes reveals a compiler warning for the three data annotations I'm focused on. Originally, they were in the System.ComponentModel.DataAnnotations namespace as part of EntityFramework.dll. But in the .NET assembly where they now live, they've moved to a sub-namespace. So I need to add one more using statement to the top of the code file:

```
using System.ComponentModel.DataAnnotations.Schema;
```

With this, the compiler is happy and so am I, because I've removed the dependency on Entity Framework in classes that have nothing to do with data access. I still have a personal aversion to putting the attributes that define database schema in my domain classes and

generally lean toward Entity Framework fluent API configurations for these tasks. However, in a small project, I find the data annotations to be convenient and quick to use.

Crossing Versions

The EF team has covered the possibility of your installing EF 4.3.x into a project that targets the .NET Framework 4.5. If you do this (whether intentionally or accidentally), a text file will be presented in the IDE that lists known issues with using EF 4.x in a .NET 4.5 project and recommends installing EF 5. Once EF 5 becomes stable and is the default package, the likelihood of developers making this mistake should disappear.

Switching to Enums, Yay!

With all of this in place I can modify my domain classes to get rid of the ugly workaround and use enums for the Rating and Experience properties. Here are the two new enums, and take note that I specified values for one but not the other so you can see how EF handles both scenarios:

```
public enum TweetRating
{
    Suxorz = 0,
    WorksForMe = 1,
    WatchOutAPIusK = 2
}

public enum TwitterExperience
{
    Newbie, BeenAround, Ninja
}
```

With the enums in place, I can modify the properties as follows:

```
[Column("ExperienceCode")]
public TwitterExperience Experience { get; set; }

[Column("RatingCode")]
public TweetRating Rating { get; set; }
```

Notice that I no longer need the attributes to specify the property range or length. More important, be aware that I'm making this change without regard for possible existing data in my demonstration database. If you want to make a change like this to an application that's in production, you'll need to prepare in advance for the data change. I'm completely altering the meaning of Experience in the database and I've also randomly changed the tweet rating range from 1-5 to 0-2.

After using Code First migrations to update the database, the Tweeter.ExperienceCode column has been changed from an nvarchar data type to an int. Both C# and Visual Basic will interpret the enum as an integer by default and will begin the enumeration with 0. Therefore, Code First will map the enum values to an int data type in the database. You can specify the enum to be a different type (within the bounds of C# and Visual Basic enums) and Code First will honor that. For example, defining an enum as long will result in properties that map to a bigint data type. But by default you'll always get an integer starting with 0. In my example, in the database,

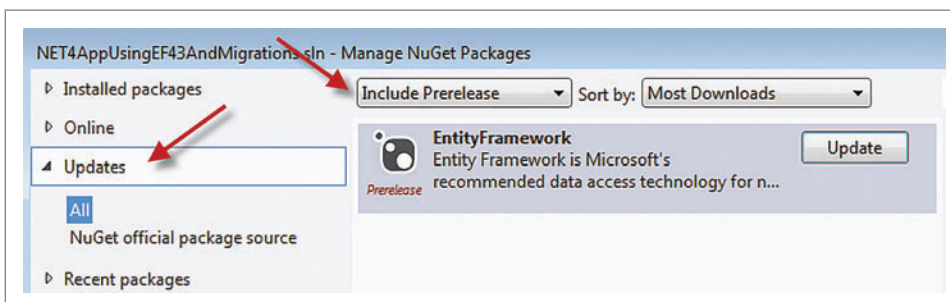


Figure 4 Finding the EntityFramework 5 Prerelease Update

Newbie will be represented by 0, BeenAround by 1 and Ninja by 2. If you think there's any chance that in the future you might want to remove any of the enum members, reorder them or add new ones other than at the end, you should assign explicit values to them as I did in the TweetRating enum. That makes it easier to change the enum without changing any of those values accidentally. Don't forget that the database will store only the numeric value, so if you *do* end up changing the value in the enum, that will effectively change the meaning of your data ... which is almost always, as C# guru Jon Skeet says, "a Bad Thing."

I like the fact that the single NuGet package for Code First and DbContext support provides compatible DLLs for both .NET 4 and .NET 4.5.

Figure 5 shows code that creates a new Tweeter instance along with a Tweet, both using the enums. After saving this data, the database shows the value of the ExperienceCode equal to 1 and Rating equal to 2.

You can use the enums in queries, and Entity Framework will take care of transforming the enum to the int value in the SQL and transforming the returned int values back to the enum values. For example, here's a LINQ query that uses an enum in the Where predicate:

```
context.Tweeters.Where(t => t.Experience ==
    TwitterExperience.Ninja)
```

In the resulting T-SQL, the Where predicate value is 2.

Smoother Move to EF 5

I'm already hearing developers express an eagerness to port their existing Entity Framework solutions to EF 5 in order to benefit from the support for enums and spatial data. Getting those projects from EF 4 to EF 5 may not be rocket science, but there were enough bumps in the road that I found the transition a bit annoying the first few times. I hope this column makes it easier for you to make the move.

I like the fact that the single NuGet package for Code First and DbContext support provides compatible DLLs for both .NET 4 and .NET 4.5. Even if I'm using an EDMX, I still start all new projects with DbContext, and therefore 100 percent of my projects now rely on the Entity Framework NuGet package.

Figure 5 Creating a New Graph of a Tweeter and a Tweet

```
var alias = new Tweeter
{
    Name = "Julie",
    UserName = "Julie",
    Bio = "Mom of Giantpuppy",
    CreateDate = DateTime.Now,
    Experience = TwitterExperience.BeenAround,
    Tweets = new List<Tweet>{new Tweet
        {
            Content = "Oh how I love that Giantpuppy",
            CreateDate = DateTime.Now,
            Rating = TweetRating.WatchOutAPlusK
        }}
};
```

Remember that EF 4 apps running on computers with the .NET Framework 4.5 installed will benefit from the performance improvements, so even if you don't get to move to Visual Studio 2012 quite yet, your users can still feel some of the love of the improvements to the Entity Framework core in the .NET Framework 4.5. ■

JULIE LERMAN is a Microsoft MVP, .NET mentor and consultant who lives in the hills of Vermont. You can find her presenting on data access and other Microsoft .NET topics at user groups and conferences around the world. She blogs at thedatafarm.com/blog and is the author of "Programming Entity Framework" (2010) as well as a Code First edition (2011) and a DbContext edition (2012), all from O'Reilly Media. Follow her on Twitter at twitter.com/julielerman.

THANKS to the following technical expert for reviewing this article: Arthur Vickers

The world leader in advanced Microsoft SQL Server
Integration Services (SSIS) tasks, components and scripts



SAS® Adapters
Reusable Scripts
USPS Address Parse
Parallel Loop
EDI Source
Table Difference
And More

CozyRoc is rare breed among technology companies

Over 100 Reusable Components



COZYROC™
Go to the next level

www.cozyroc.com
sales@cozyroc.com
(919) 249-7421



Humongous Windows Azure

Personally, I love the way things cycle. It always seems to me that each evolution of an object or a mechanism expresses a duality of purpose that both advances and restates a position of the past. Technology is a great place to see this, because the pace at which changes have been taking place makes it easy to see a lot of evolutions over short periods of time.

For me, the NoSQL movement is just such an evolution. At first we had documents, and we kept them in files and in file cabinets and, eventually, in file shares. It was a natural state of affairs. The problem with nature is that at scale we can't really wrap our brains around it. So we rationalized the content and opted for rationalized and normalized data models that would help us predictably consume space, store data, index the data and be able to find it. The problem with rational models is that they're not natural.

Enter NoSQL, a seeming mix of the natural and relational models. NoSQL is a database management system optimized for storing and retrieving large quantities of data. It's a way for us to keep document-style data and still take advantage of some features found in everyday relational database management systems (RDBMSes).

One of the major tools of NoSQL is MongoDB from 10gen Inc., a document-oriented, open source NoSQL database system, and this month I'm going to focus on some of the design and implementation aspects of using MongoDB in a Windows Azure environment. I'm going to assume you know something about NoSQL and MongoDB. If not, you might want to take a look at Julie Lerman's November 2011 Data Points column, "What the Heck Are Document Databases?" (msdn.microsoft.com/magazine/hh547103), and Ted Neward's May 2010 The Working Programmer column, "Going NoSQL with MongoDB" (msdn.microsoft.com/magazine/ee310029).

First Things First

If you're thinking of trying out MongoDB or considering it as an alternative to Windows Azure SQL Database or Windows Azure Tables, you need to be aware of some issues on the design and planning side, some related to infrastructure and some to development.

Deployment Architecture

Generally, the data back end needs to be available and durable. To do this with MongoDB, you use a replication set. Replication sets provide both failover and replication, using a little bit of artificial intelligence (AI) to resolve any tie in electing the primary node of the set. What this means for your Windows Azure roles is that you'll need three instances to set up a minimal replication set, plus a storage location you can map to a drive for each of those roles. Be aware that due to

differences in virtual machines (VMs), you'll likely want to have at least midsize VMs for any significant deployment. Otherwise, the memory or CPU could quickly become a bottleneck.

Figure 1 depicts a typical architecture for deploying a minimal MongoDB ReplicaSet that's not exposed to the public. You could convert it to expose the data store externally, but it's better to do that via a service layer. One of the problems that MongoDB can help address via its built-in features is designing and deploying a distributed data architecture. MongoDB has a full feature set to support sharding; combine that feature with ReplicaSets and Windows Azure Compute and you have a data store that's highly scalable, distributed and reliable. To help get you started, 10gen provides a sample solution that sets up a minimal ReplicaSet. You'll find the information at bit.ly/NZROWJ and you can grab the files from GitHub at bit.ly/L6cqMF.

Data Schema

Being a wiz at DB schema design may actually hinder you when designing for a NoSQL approach. The skills required are more like object modeling and integration design for messaging infrastructures. There are two reasons for this:

1. The data is viewed as a document and sometimes contains nested objects or documents.
2. There's minimal support for joins, so you have to balance the storage format of the data against the implications of nesting and the number of calls the client has to make to get a single view.

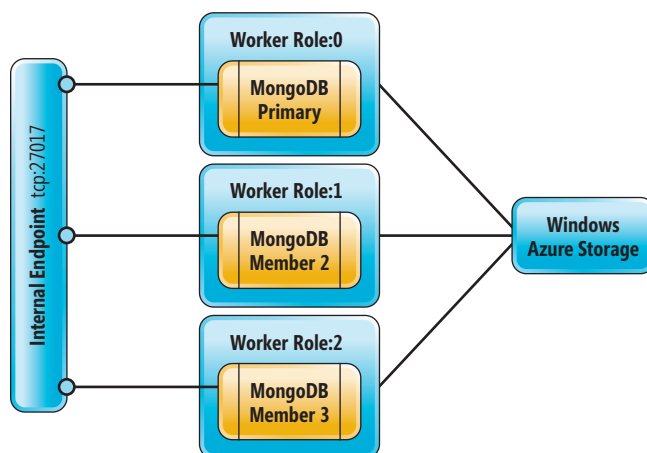


Figure 1 Windows Azure MongoDB Deployment

Raising the Bar...

And Pie, and Spline and Scatter... on Mobile Business Intelligence



Visualize your BI in 50+ ways, and on just as many devices. From candlesticks to polar and radial charts, nobody offers the breadth and depth of dynamic, high fidelity, totally mobile charting solutions that you get from Infragistics' NetAdvantage. **Try a free, fully supported trial of NetAdvantage for .NET today!**

www.infragistics.com/NET



Infragistics Sales US 800 231 8588 • Europe +44 (0) 800 298 9055 • India +91 80 4151 8042 • APAC (+61) 3 9982 4545

Copyright 1996-2012 Infragistics, Inc. All rights reserved. Infragistics and NetAdvantage are registered trademarks of Infragistics, Inc. The Infragistics logo is a trademark of Infragistics, Inc. All other trademarks or registered trademarks are the respective property of their owners.





Figure 2 Direct Schema Translation

One of the first activities of moving from a relational mindset to the MongoDB document perspective is redesigning the data schema. For some objects that are separate in a relational model, the separation is maintained. For example, Products and Orders will still be separate schema in MongoDB, and you'll still use a foreign key to do lookups between the two. Oversimplifying a bit, the redesign for these two objects in relation to one another is mostly straightforward, as shown in **Figure 2**.

However, it may not be as easy when you work with schemas that aren't as cleanly separated conceptually, even though they may be easily and obviously separated in a relational model. For example, Customers and CustomerAddresses are entities that might be merged such that Customer will contain a collection of associated addresses (see **Figure 3**).

You'll need to take a careful look at your relational model and consider every foreign key relationship and how that will get represented in the entity graph as it's translated to the NoSQL model.

Data Interaction

Both query behavior and caching behavior are important in a relational system, but it's caching behavior that remains most important here. Much as with Windows Azure Tables, it's easy to drop an object into MongoDB. And unlike Windows Azure Tables and more like Windows Azure SQL Databases, any of the fields can be indexed, which allows for better query performance on single objects. However, the lack of joins (and general lack of query expressiveness) turns what could once be a query with one or more joins for a chunky data return into multiple calls to the back-end data store to fetch that same data. This can be a little daunting if you want to fetch a collection of objects and then fetch a related collection for each item in the first collection. So, using my relational pubs database, I might write a SQL query that looks something like the following to fetch all author last names and all titles from each author:

```
Select authors.au_name, authors.au_id,
       titles.title_id, titles.title
From authors inner join titleauthor
  on authors.au_id = titleauthor.au_id
 inner join titles on
  titles.title_id = titleauthor.title_id
Order By authors.au_name
```

In contrast, to get the same data using the C# driver and MongoDB, the code looks like what's shown in **Figure 4**.

There are ways you might optimize this through code and structure, but don't miss the point that while MongoDB is well-suited for direct queries even on nested objects, more complex queries that require cross-entity sets are a good bit more ... well, let's just say more manual. Most of us use LINQ to help bridge the object-to-relational world. The interesting thing with MongoDB

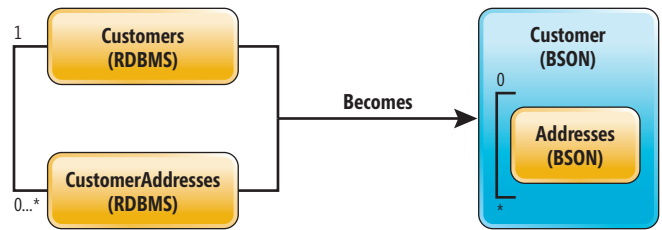


Figure 3 Converting Relational Schema to Nested Object Schema

is that you'll want that bridge, but for the opposite reason—you'll miss the relational functionality.

You might also miss referential constraints, especially foreign key constraints. Because you can literally add anything into the MongoDB collection, an item may or may not have the proper data to relate it to other entities. While this might seem like a failing of the platform if you're a die-hard RDBMS fan, it isn't. It is, in fact, a departure in philosophy. For NoSQL databases in general, the idea is to move the intelligence in the system out of the data store and let the data store focus on the reading and writing of data. Thus, if you feel the need to explicitly enforce things like foreign key constraints in your MongoDB implementation, you'll do that through the business or service layer that sits in front of the data store.

The Migration

Once you've redesigned the data schemas and considered query behavior and requirements it's time to get some data out there in the cloud in order to work with it.

The bad news is that there's no wizard that lets you point to your Windows Azure SQL Database instance and your MongoDB

Figure 4 Joining to MongoDB Collections

```
MongoDatabase mongoPubs = _mongoServer.GetDatabase("Pubs");
MongoCollection<BsonDocument> authorsCollection =
    mongoPubs.GetCollection("Authors");
MongoCursor<BsonDocument> authors = authorsCollection.FindAll();
string auIdQueryString = default(string);
Dictionary<string, BsonDocument> authorTitles =
    new Dictionary<string, BsonDocument>();

// Build string for "In" comparison
// Build list of author documents, add titles next
foreach (BsonDocument bsonAuthor in authors)
{
    auIdQueryString = bsonAuthor["au_id"].ToString() + ",";
    authorTitles.Add(bsonAuthor["au_id"].ToString(), new BsonDocument{["au_id",
        bsonAuthor["au_id"].ToString()],
        ["au_name", bsonAuthor["au_name"]]});
    authorTitles.Add("titles", new BsonDocument(new
        Dictionary<string, object>()));
}

// Adjust last character
auIdQueryString = auIdQueryString.Remove(auIdQueryString.Length-1,1);
// Create query
QueryComplete titleByAu_idQuery = Query.In("au_id", auIdQueryString);
Dictionary<string, BsonDocument> bsonTitlesToAdd =
    new Dictionary<string, BsonDocument>();
// Execute query, coalesce authors and titles
foreach (BsonDocument bsonTitle in authorsCollection.Find(titleByAu_idQuery))
{
    Debug.WriteLine(bsonTitle.ToJson());
    // Add to author BsonDocument
    BsonDocument authorTitlesDoc = authorTitles[bsonTitle["au_id"].ToString()];
    ((Dictionary<string, object>) authorTitlesDoc["titles"]).
        Add(bsonTitle["title_id"].ToString(), bsonTitle);
}
```

MetroTactual

[me-troh tak-choo-uhl]



Product ID	Name	Product Number	Color
1	Adjustable Race	AR-5381	
Product ID	Address ID	Shelf	Bin
1	1	A	1
1	6	B	5
1	50	A	5
Count = 3.00	Count = 3.00	Count = 3.00	Count = 3.00
Min = 1.00	Min = 1.00	Min = 324.00	Min = 408.00
Max = 1.00	Max = 50.00	Max = 408.00	Max = 1085.00
Sum = 3.00	Sum = 57.00	Sum = 1085.00	Sum = 361.67
AVG = 1.00	AVG = 19.00		



Compatible with
Microsoft® Visual Studio® 2012

noun, adjective

1. Modern, clean, sleek, stylish, touch-friendly design and UX
 2. Feng Shui for your apps
 3. Available in NetAdvantage 12.1 toolsets
- See also: NetAdvantage for .NET

Try your free, fully supported trial today.
www.infragistics.com/.NET



Infragistics Sales US 800 231 8588 • Europe +44 (0) 800 298 9055 • India +91 80 4151 8042 • APAC (+61) 3 9982 4545

Copyright 1996-2012 Infragistics, Inc. All rights reserved. Infragistics and NetAdvantage are registered trademarks of Infragistics, Inc. The Infragistics logo is a trademark of Infragistics, Inc. All other trademarks or registered trademarks are the respective property of their owners.



Figure 5 Migrating Data with LINQ and MongoDB

```
pubsEntities myPubsEntities = new pubsEntities();
var pubsAuthors = from row in myPubsEntities.authors
    select row;
MongoDatabase mongoPubs = _mongoServer.GetDatabase("Pubs");
mongoPubs.CreateCollection("Authors");
MongoCollection<BsonDocument> authorsCollection =
    mongoPubs.GetCollection("Authors");
BsonDocument bsonAuthor;

foreach (author pubAuthor in pubsAuthors)
{
    bsonAuthor = pubAuthor.ToBsonDocument();
    authorsCollection.Insert(bsonAuthor);
}
```

instance and click Migrate. You'll need to write some scripts, either in the shell or in code. Fortunately, if code for the MongoDB side of the equation is constructed well, you'll be able to reuse a good portion of it for normal runtime operation of the solution.

The first step is referencing the MongoDB.Bson and MongoDB.Driver libraries and adding the using statements:

```
using MongoDB.Bson.IO;
using MongoDB.Bson.Serialization;
using MongoDB.Bson.Serialization.Attributes;
using MongoDB.Bson.Serialization.Conventions;
using MongoDB.Bson.Serialization.IdGenerators;
using MongoDB.Bson.Serialization.Options;
using MongoDB.Bson.Serialization.Serializers;
using MongoDB.Driver.Builders;
using MongoDB.Driver.GridFS;
using MongoDB.Driver.Wrappers;
```

Objects will then show some new methods on them that are extremely useful when you're trying to move from regular .NET objects to the Bson objects used with MongoDB. As **Figure 5** shows, this becomes quite obvious in a function for converting the output rows from a database fetch into a BsonDocument to save into MongoDB.

The simple example in **Figure 5** converts the data directly using the MongoDB extension methods. However, you have to be careful, especially with LINQ, when performing this type of operation. For example, if I attempt the same operation directly for Titles, the depth of the object graph of the Titles table in the entity model will cause the MongoDB driver to produce a stack overflow error. In such a case, the conversion will be a little more verbose in code, as shown in **Figure 6**.

To keep the conversion as simple as possible, the best approach is to write the SQL queries to return individual entities that can

Figure 6 Converting Values Individually

```
pubsEntities myPubsEntities = new pubsEntities();
var pubsTitles = from row in myPubsEntities.titles
    select row;
MongoDatabase mongoPubs = _mongoServer.GetDatabase("Pubs");
MongoCollection<BsonDocument> titlesCollection =
    mongoPubs.GetCollection("Titles");
BsonDocument bsonTitle;

foreach (title pubTitle in pubsTitles)
{
    bsonTitle = new BsonDocument(
        {"titleId", pubTitle.title_id},
        {"pub_id", pubTitle.pub_id},
        {"publisher", pubTitle.publisher.pub_name},
        {"price", pubTitle.price.ToString()},
        {"title1", pubTitle.title1});
    titlesCollection.Insert(bsonTitle);
}
```

more easily be added to the appropriate MongoDB collection. For BsonDocuments that have child document collections, it will take a multistep approach to create the parent BsonDocument, add the child BsonDocuments to the parent BsonDocument, and then add the parent to the collection.

The obvious bits you'll need to convert if moving from a Windows Azure SQL Database to a MongoDB implementation is all of the code that lives in stored procedures, views and triggers. In many cases, the code will be somewhat simpler, because you'll be dealing with one BsonDocument with children that you persist in its entirety instead of having to work across the relational constraints of multiple tables. Furthermore, instead of writing TSQL, you get to use your favorite .NET language, with all the support of Visual Studio as the IDE. The code that may not be initially accounted for is what you'll have to create to be able to do transactions across documents. In one sense, it's a pain to have to move all of that Windows Azure SQL Database platform functionality into application code. On the other hand, once you're done you'll have an extremely fast and scalable data back end, because it's focused solely on data shuttling. You also get a highly scalable middle tier by moving all of that logic previously trapped in the RDBMS into a proper middle-tier layer.

One last note of some significance is that, due to the nature of the data store, the data size will likely increase. This is because every document has to hold both schema and data. While this may not be terribly important for most, due to the low cost of space in Windows Azure Tables, it's still something that needs to be accounted for in the design.

Final Thoughts

Once the data is available in MongoDB, working with it will, in many regards, feel familiar.

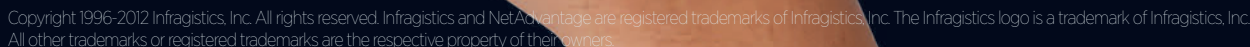
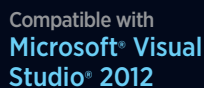
As of C# driver 1.4 (currently on 1.5.0.4566) the LINQ support is greatly improved, so writing the code won't feel completely unfamiliar. So, if your project or solution might benefit from a NoSQL data store like MongoDB, don't let the syntax frighten you, because the adjustment will be minimal. Keep in mind, however, that there are some important differences between a mature, robust RDBMS platform—such as Windows Azure SQL Database—and MongoDB. For example, health and monitoring will require more manual work. Instead of monitoring only some number of Windows Azure SQL Database instances, you'll have to monitor the host worker roles, the Windows Azure Blob storage host of the database files and the log files of MongoDB itself.

NoSQL solutions offer great performance for some database operations, and some useful and interesting features that can really be a boon to a solution development team. If you have a large amount of data and you're on a limited budget, the MongoDB on Windows Azure option might be a great addition to your solution architecture. ■

JOSEPH FULTZ is a software architect at Hewlett-Packard Co., working as part of the HP.com Global IT group. Previously he was a software architect for Microsoft, working with its top-tier enterprise and ISV customers to define architecture and design solutions.

THANKS to the following technical expert for reviewing this article:
Wen-ming Ye

www.infragistics.com/NET



A More Productive IDE for Modern Applications

Peter Vogel

While **Visual Studio 2012 Professional** supports several new technologies (for example, Windows Store apps and the ASP.NET Web API), the Visual Studio team seems to have taken the opportunity to concentrate on creating a “better IDE.” They started with an overhaul of the UI. Although the Visual Studio 2012 UI overhaul isn’t nearly as drastic as what Microsoft has done with Windows Store apps, it includes many significant changes. That’s not to say there aren’t lots of non-UI improvements, but let’s talk first about the “why” of those changes.

The Visual Studio 2012 “better IDE” focuses on three goals to help you be more productive: reducing clutter, simplifying common tasks and improving usability. While the new monochromatic UI and use of “all caps” in top-level menus have attracted the most attention, other, more significant changes are going unnoticed. For example, the number of default toolbars that appear as Visual Studio changes from one mode to another is sharply reduced in Visual Studio 2012 (see **Figure 1**). While this reduces clutter, the more practical

result is room for a couple more lines of code on the screen. This reduces the need to scroll up and down and allows developers to more easily see “the whole method,” simplifying a very common task: writing code. It’s a change that might seem trivial, but it reflects the direction of many of the changes in Visual Studio 2012.

Accessing Code

Access to code is a key feature of the new Visual Studio 2012 experience. For example, if you have the preview button selected, your first reaction when clicking on a file in Solution Explorer might be that Visual Studio 2012 is now opening files with single-clicks rather than double-clicks. However, what you’re seeing when you single-click on a code file is a preview of the file (a visual clue is that the window’s tab appears on the right-hand side of the tab well). Single-clicking on another file in Solution Explorer dismisses the existing preview and previews the new file. If, however, you make a change to a previewed file, the tab shifts to the left and the file stays open in edit mode.

But searching for code by clicking on each file is inefficient—the new search/filter options at the top of some of the tool windows provide a much more effective way to find what you want. The most obvious example is the Search in Solution textbox at the top of Solution Explorer. It lets you look for file names and—more important—member names in any project in the solution (though not with ASP.NET Web site projects). You no longer need to open a new window to search for text. The result is less clutter and simplification of a common task.

Also reflecting the move to reduce clutter while improving usability, Solution Explorer now combines, along with the Find window, features of Class View with the traditional Solution Explorer file-based view to let you drill down into individual members within your files (see **Figure 2**). Complementing this feature is the new Home button at the top of Solution Explorer that restores Solution Explorer to the standard list of files.

This article discusses a prerelease version of Visual Studio 2012. All related information is subject to change.

This article discusses:

- Accessing code
- IntelliSense improvements
- ASP.NET and ASP.NET MVC development
- Using Page Inspector
- Testing
- Building Windows Store apps
- Using Blend for Visual Studio

Technologies discussed:

Visual Studio 2012

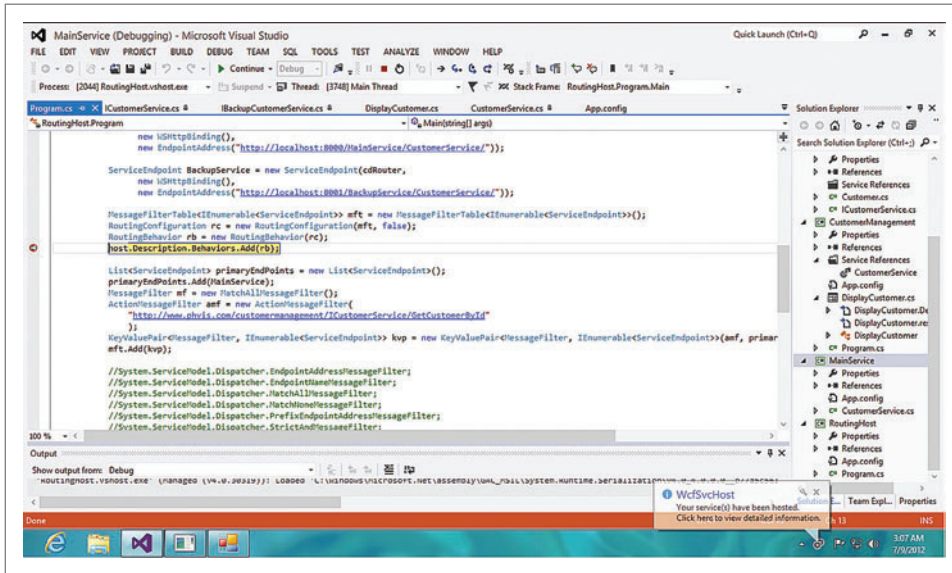


Figure 1 Reduced Color and Chrome Make Code Highlights Stand out More

Quick Launch (which appears at the right-hand side of the menu bar) seems less useful to me. It searches for Visual Studio assets rather than project assets. If you're looking for menus or Visual Studio options by keyword, Quick Launch will let you find the matching Visual Studio item and then go directly to it. It's not clear to me how often I'll want to do that, but Quick Launch might encourage me to do it more.

IntelliSense Expands

IntelliSense continues to expand its search parameters to make it easier for you to find classes and members. As you type, IntelliSense not only matches any part of a word but also selectively matches any uppercase letters in the names of classes and members. In an .aspx file, for example, typing "oc" brings up an IntelliSense list showing OutputCache.

options for generating IntelliSense support for overloaded functions. Visual Studio 2012 even takes a stab at providing IntelliSense for dynamically loaded script files. Selecting a JavaScript function call and pressing F12 (or selecting Go to Definition from a context menu) will take you to the file containing the function (except for generated code). Not all JavaScript support is equally useful, though—when Visual Studio 2012 can't determine the data type for a variable, IntelliSense often just lists every JavaScript entity available.

ASP.NET and ASP.NET MVC Developers

ASP.NET MVC developers will appreciate the new project templates that support creating mobile apps (including the jQuery mobile libraries) and apps that support the ASP.NET Web API. Another project type supports the Microsoft client-side-enabled Single Page Application, which integrates several JavaScript APIs—the Knockout Model-View-View-Model (MVVM)/data-binding library, the HTML5 History API and the Microsoft Upshot library for managing downloaded objects—to support creating AJAX apps. If you're happy with the Microsoft choice of JavaScript libraries—and they are good choices—you won't have to assemble your own technology set from scratch any more.

Visual Studio 2012 also will now auto-format JavaScript code for Web and SharePoint developers. If you've got a different standard for JavaScript formatting, you'll either need to turn this feature off in Tools | Options or adjust your expectations.

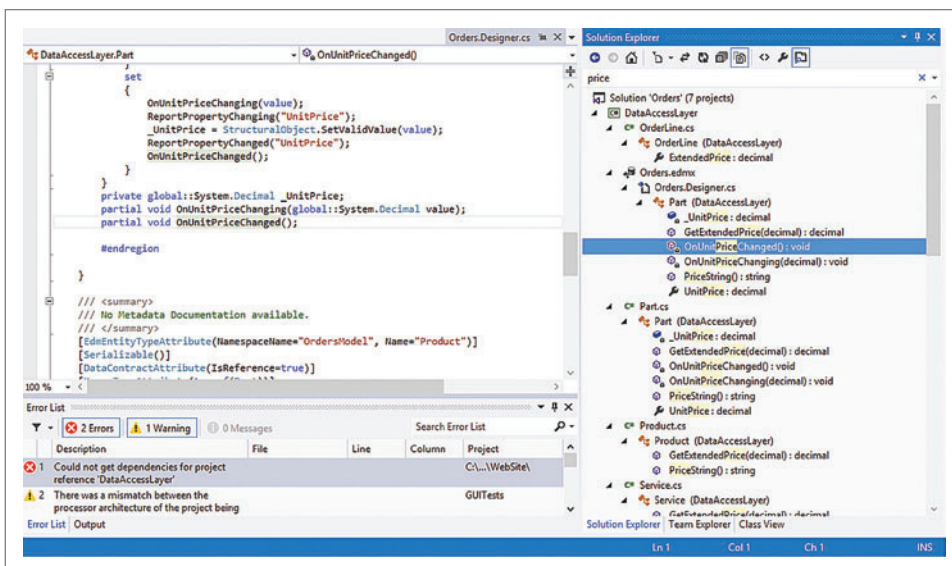


Figure 2 Solution Explorer Now Acts as a Kind of Object Browser

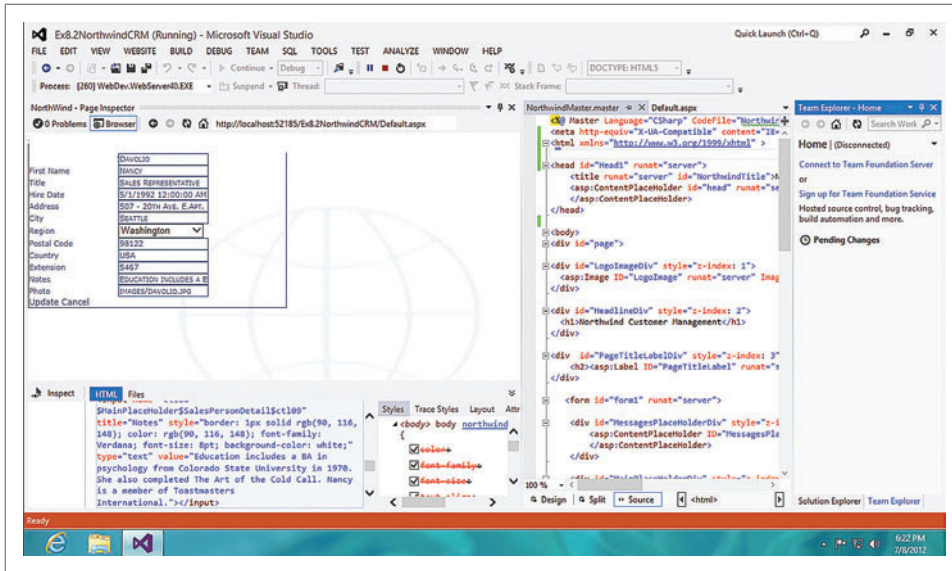


Figure 3 Page Inspector Shows All the Markup that Controls How the Page Is Rendered

Visual Studio 2012 supports the new HTML5 tags—the ASP.NET default.aspx page even includes section tags. Developers can stop coercing `` and `<div>` tags using shared CSS classes as a way of identifying their page's structure and start using tags dedicated to that task. IntelliSense for older HTML tags supports the new HTML5 attributes (including the custom data attributes and the Accessible Rich Internet Applications accessibility attributes). Of course, you have to count on the browser recognizing these new tags and attributes.

Page Inspector

The major change for Web developers comes in debugging. You can still press F5 to debug your ASP.NET and ASP.NET MVC applications, but a new dropdown list in the debug toolbar makes it easy to switch between browsers by automatically listing all the browsers installed on your computer. The real jewel in the crown on this list, however, is Page Inspector, which will change the way Web developers solve problems in their pages.

The typical debugging process for an ASP.NET page has seven steps: Open the page in the browser, see something wrong, guess at the problem, shut down the browser, apply your fix, open the page in the browser (again) and see if the problem is gone. Page Inspector short-circuits this whole process; just right-click on a page (.aspx, .cshtml or .vbhtml) in Solution Explorer and select View in Page Inspector. Visual Studio 2012 then reconfigures itself into several new panes, showing your page as it's rendered in the browser along with the source file that you selected, a tree view of the HTML sent to the browser and an interactive view of the CSS applied to your page—and they're all live (see Figure 3).

As you move your cursor from place to place in the browser view, the HTML and CSS panes update to show you what's contributing to what you see in the browser. While your source file doesn't move to match your browser, your source does remain updateable. If you make a change to your source, the browser view refreshes to exhibit the results of your change. In the CSS view, you can disable rules or, by clicking on a rule, switch to the CSS file containing the rule. Once there, if you change the rule, the page redisplayes to show the result of your change.

It's not a perfect solution—the view is crowded (even on a 17-inch monitor) and Page Inspector seems to have trouble with absolute

positioning in stylesheets (but you probably shouldn't be using absolute positioning, anyway). Even with those caveats, Page Inspector is going to be a tool that I won't be able to live without.

Testing

While Web developers get improvements in debugging, everyone benefits from the enhanced support for test-driven development (TDD). The old test output window has been replaced by a new Test Explorer (see Figure 4), available from Test | Windows, which centralizes most TDD activities. Test Explorer lists the status of all of your tests from their last test run, including the time each test took (giving you an early look at where your performance bottlenecks might be). Clicking on a failed test in Test Explorer displays its information at the bottom of Test Explorer. Double-clicking on a test in Test Explorer takes you to its code so you can start a debugging

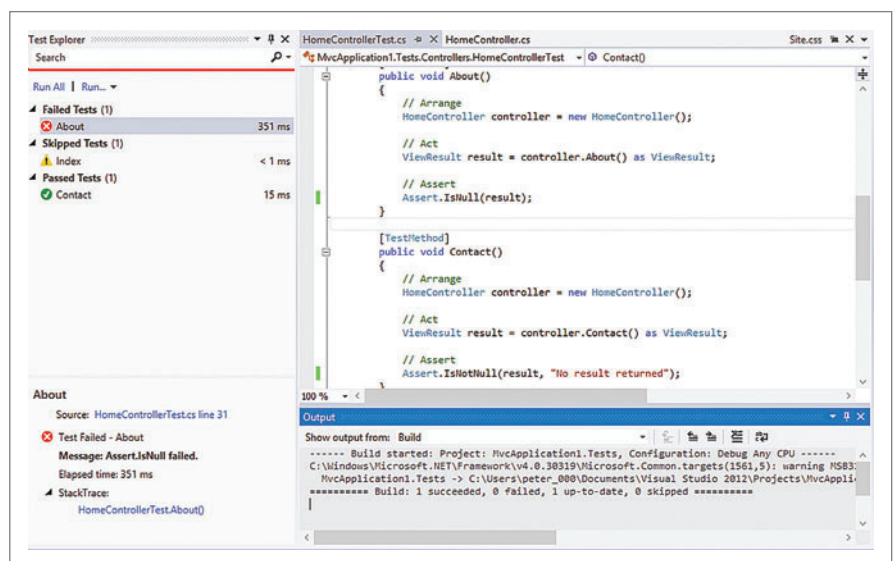


Figure 4 Test Explorer Provides a More Interactive Way to View and Run Tests

We didn't invent the Internet...

...but our components help you power the apps that bring it to business.



TOOLS • COMPONENTS • ENTERPRISE ADAPTERS

- **E-Business**
AS2, EDI/X12, NAESB, OFTP ...
- **Credit Card Processing**
Authorize.Net, TSYS, FDMS ...
- **Shipping & Tracking**
FedEx, UPS, USPS ...
- **Accounting & Banking**
QuickBooks, OFX ...
- **Internet Business**
Amazon, eBay, PayPal ...
- **Internet Protocols**
FTP, SMTP, IMAP, POP, WebDav ...
- **Secure Connectivity**
SSH, SFTP, SSL, Certificates ...
- **Secure Email**
S/MIME, OpenPGP ...
- **Network Management**
SNMP, MIB, LDAP, Monitoring ...
- **Compression & Encryption**
Zip, Gzip, Jar, AES ...



The Market Leader in Internet Communications, Security, & E-Business Components

Each day, as you click around the Web or use any connected application, chances are that directly or indirectly some bits are flowing through applications that use our components, on a server, on a device, or right on your desktop. It's your code and our code working together to move data, information, and business. We give you the most robust suite of components for adding Internet Communications, Security, and E-Business Connectivity to

any application, on any platform, anywhere, and you do the rest. Since 1994, we have had one goal: to provide the very best connectivity solutions for our professional developer customers. With more than 100,000 developers worldwide using our software and millions of installations in almost every Fortune 500 and Global 2000 company, our business is to connect business, one application at a time.

connectivity
powered by 

To learn more please visit our website →

www.nsoftware.com

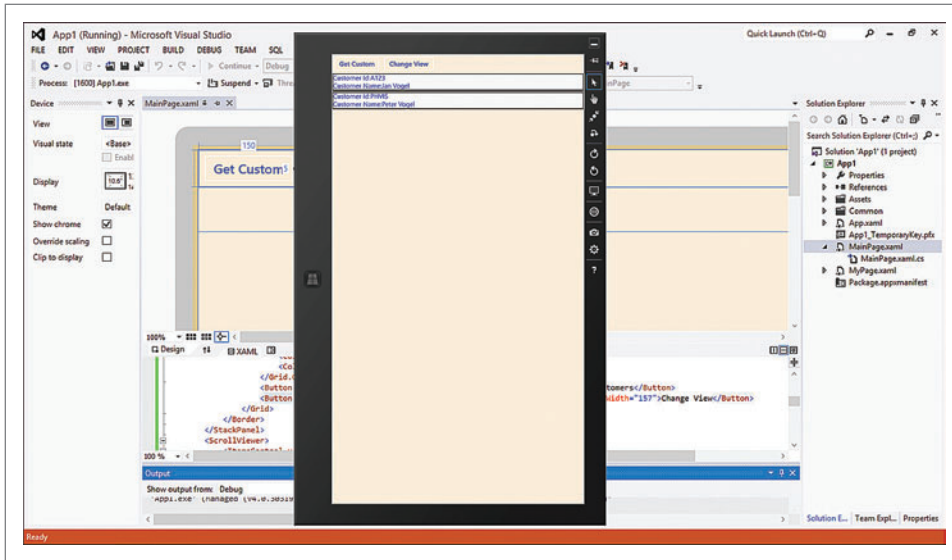


Figure 5 The Simulator Option for Testing Windows Store Applications Mimics a Tablet

session with it. If the list of tests gets very long, you can filter it by keyword. Test Explorer now groups all the failed tests together instead of intermixing them with all the other tests, and shows all tests marked with the Ignore attribute instead of silently skipping them.

Test Explorer also offers a new option for running tests: to run all tests that haven't been run yet. Initially, your tests all start in the Not Run category and, as you run those tests that prove your new code works, some tests shift (eventually) into the Passed category. Once you've proven that your new code works, you can pick Run Not Run Tests to run whatever tests remain un-run to prove that you haven't introduced any new errors to your old code. Test Explorer is another example of simplifying common tasks while reducing visual clutter in the UI.

The most significant change in testing is, unfortunately, only available in Visual Studio 2012 Ultimate: automatically executing your tests after every build. Ultimate and Premium also include support for test code coverage (allowing you to see which lines of code haven't been tested), but that's a tool of doubtful usefulness, at best, in my opinion (some professional developers disagree).

And, of Course, Windows 8

Not surprisingly, Microsoft has provided good support for developers creating applications for Windows 8. You'll need a Windows 8 developer license to create these apps, but Visual Studio 2012 throws you into the wizard that walks you through the process of getting the license the first time you select a this project type.

Visual Studio 2012 provides six project types for Windows Store applications: three UIs (blank, grid and split); a class library; a language-independent Windows Runtime Component; and a test library. The variety of project types suggests that Windows Store applications live in their own world.

Like the Web application support for multiple browsers, you can debug Windows Store apps in a variety of environments by selecting the environment you want from a dropdown list on the toolbar. The default environment is "Local Machine," but you

can also select Simulator, which brings up the Microsoft tablet-like simulator, as shown in **Figure 5**.

From the right-hand side of the simulator, buttons turn your mouse into various kinds of pointers that mimic touch interactions with your application (assuming that you don't have a touchscreen on your development machine). Another button lets you set the location for your simulated tablet using latitude, longitude and elevation for testing location-based applications. The Rotate Clockwise and Rotate CounterClockwise buttons let you swing the tablet between portrait and landscape modes. There's even a button for capturing a screenshot of your simulated tablet that drops

the screenshot into the Windows clipboard. Using the simulator isn't the same as testing on a real physical device, but it's good enough for initial testing and will save you having to shoot your app over to another device just to run a simple test.

Rethinking the UI

Much has been said about the reduction in "visual clutter" in the Visual Studio 2012 UI—so much that Microsoft added some "color clutter" between the beta and the release candidate (RC). In general, the goal in any UI design is to make similar things look the same (that is, all menu items should look alike, as should all buttons) and different things look different (that is, buttons and menu items should look different). The Visual Studio 2012 RC seems to have struck a "good enough" balance in making things look appropriately (and obviously) the same and different. It will be interesting to see if, in a few years, developers start thinking of earlier versions of Visual Studio as "gaudy."

While the UI color scheme attracted some discussion, more has been said about the Microsoft decision to put the top-level menu items in all caps to make the menu stand out from the clutter of toolbars. Several studies have shown that adult readers recognize words, in part, by the shape of their ascenders (d, b and f) and descenders (g and y) when the words are printed in mixed case. In all caps, though, all words have the same shape: rectangular. In Visual Studio 2012, however, you can't really say that what developers do when accessing a menu is "reading." Because the position of the menu items is fixed (File, Edit, View, Window, Help), developers probably find menus as much by position as by the item's actual text.

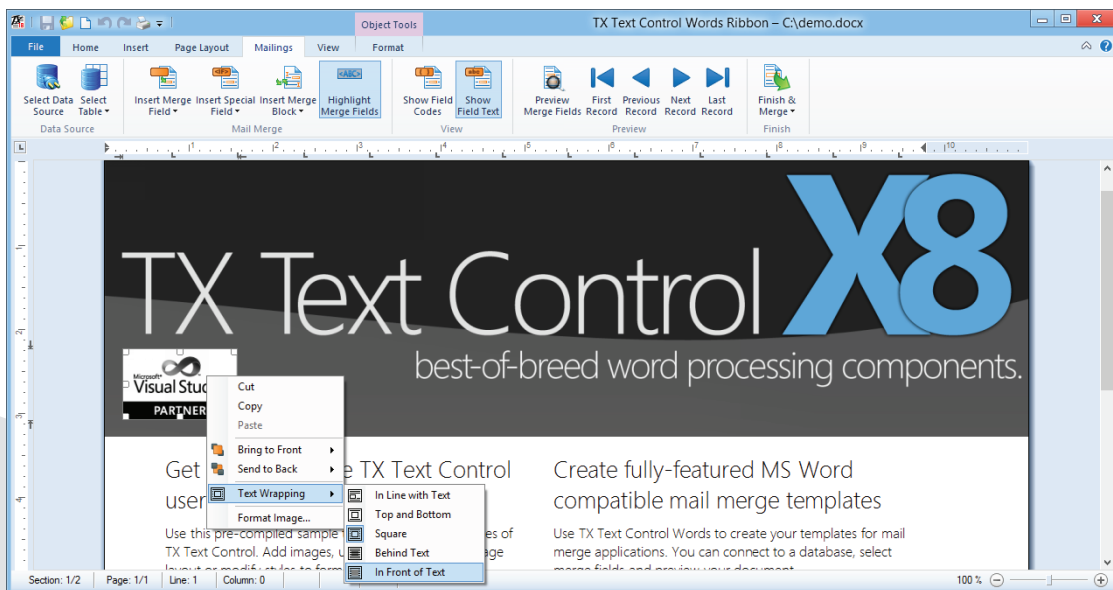
Experienced users are always disconcerted by any change, but the real impact of all caps will be on new users. Does the reduction in individual menu item recognizability pay off in increased "mouse marksmanship"?

In the meantime, various registry hacks and Visual Studio 2012 extensions have been posted that allow developers to have the top-level menu items display in mixed case. An option in the final release version of Visual Studio 2012 will probably make everyone happy.

WORD PROCESSING COMPONENTS

① Rich Text Editing

Integrate professional rich text editing into your .NET based applications.



PDF Reflow - Open PDF Documents

Load, view, modify and convert Adobe PDF documents and reuse formatted text. Search and analyze documents such as invoices or delivery notes.

Spell Checking

Add the fastest spell checking engine to your Windows Forms, WPF or ASP.NET applications.

Reporting Redefined

Build MS Word compatible mail merge applications with Master-Detail views.

Free License

Download our 100% free Express version.



Blend for Visual Studio ... and Windows Store Apps

Part of the support for creating Windows Store applications includes bundling Blend for Visual Studio with the Visual Studio 2012 package. However, Blend is an optional part, so you'll need to pick the Custom Installation option to include it in your Visual Studio 2012 installation. Perversely, while Blend supports both XAML and Windows Presentation Foundation (WPF), it's not available from within Visual Studio 2012 for anything but Windows Store application development.

Although Blend is shipped with Visual Studio 2012, you can't really say that it's integrated with Visual Studio 2012. Right-clicking on a UI file in a Windows Store application allows you to pick the Open in Blend option. Selecting that option opens Blend in a new window. Fortunately, Blend picks up your project's file list so you can modify your files without having to return to Visual Studio 2012 (in fact, when you do return to Visual Studio 2012, you'll get a "File has been updated outside of Visual Studio" message for the files changed by Blend). Files added to your project in Blend are added to your Visual Studio 2012 project. But you still need to be careful as you move between the two windows: You can switch to a different solution in Visual Studio 2012 without Blend noticing or caring. Having said all that, Blend lets you do a great many things graphically that would otherwise force you into working directly with XAML.

More Goodies: Multiple Monitors, Debugging, Windows Azure and SharePoint

There's more, of course. You can disengage windows from Visual Studio 2012 and then dock them to each other (Microsoft calls this a "raft"). You can then drag your raft to the position—or monitor—of your choice. The debugging windows are much more thread-aware and, more usefully, let you flag the threads you're interested in and limit the display to those threads.

Windows Azure developers get better integration between the Windows Azure Platform Management Portal and Visual Studio 2012. Moving an ASP.NET MVC app to the cloud requires only a few clicks, and you never have to exit Visual Studio to use the Windows Azure management portal. Not everything that Windows Azure developers need is in Visual Studio 2012 Professional, however. If you want to copy a database to the cloud (either just the schema or the schema and data), you'll need to download the SQL Azure Migration Wizard from CodePlex (bit.ly/bYN8Vb).

SharePoint developers get new templates for site columns and content types—fundamental building blocks for SharePoint sites. Visual Studio 2012 also, finally, gets a visual list designer for SharePoint that's almost as good as the one in SharePoint. SharePoint developers can also publish their solutions directly from Visual Studio 2012 to a remote site (which will drive whoever is supposed to be managing releases crazy).

While WPF and Silverlight developers don't get the minimal integration with Blend that Windows Store app developers do, some Blend menu choices (such as Pin Active Container and Create Data Bindings For) have been added to the XAML designer's context menus.

Upgrade?

Since Visual Studio 2008, Visual Studio has been "disconnected" from the corresponding version of the Microsoft .NET Framework,

Team Foundation Service

As software best practices continue to encourage creating lots of simple, dedicated objects that are assembled to create complex applications, managing the multiplicity of software components through their entire lifecycle has become increasingly important. The home for application lifecycle management (ALM) tools in the Microsoft universe is Visual Studio Team Foundation Server (TFS). However, especially in smaller teams, there's an assumption (sometimes misplaced) that the costs of installing, configuring and managing TFS would wipe out whatever ALM benefits it would provide. This is where Team Foundation Service comes in: It's TFS in the cloud.

While Team Foundation Service is still in preview mode, you can experiment with it (or just review its features) at tfspreview.com. Team Foundation Service integrates with both Visual Studio 2012 and Eclipse (though not all features are available for Eclipse).

With Team Foundation Service, you can use PowerPoint at the start of your application development process to storyboard your application. Once development begins, the service provides source control, continuous unit testing (though only on check-in), and tools for managing information about features, task, bugs, feedback and backlog. Setting up continuous integration builds is relatively easy (at least for simple projects) and includes automatic deployment to Windows Azure. Team Explorer, a Web-based tool, lets you review your project from anywhere.

Pricing still wasn't announced at the time this article was written. Microsoft has said there will continue to be a free level of the service after it leaves preview mode. However, Microsoft has also said that there will be paid levels of the service for users who aren't satisfied with the free level.

meaning you could upgrade to the new IDE without upgrading your version of the framework. You have even more freedom in Visual Studio 2012 because all projects are not automatically upgraded to the .NET Framework 4.5 with no way to go back. You can successfully round-trip a project between Visual Studio 2012 and 2010, provided that you don't use SQL Server 2012 Express LocalDB in Visual Studio 2012 (a zero-configuration version of SQL Server Express only supported in Visual Studio 2012) or some feature only available in the .NET Framework 4.5.

If you're going to upgrade to the .NET Framework 4.5, of course, you have no choice but to upgrade to Visual Studio 2012. But if you're not upgrading to the .NET Framework 4.5, Visual Studio 2012 is still worth considering. The price isn't unreasonable (Professional is \$499 without an MSDN subscription) and, for ASP.NET developers, Page Inspector is probably worth the cost all by itself. Windows Azure developers will appreciate the deeper integration with Visual Studio 2012. The new Solution Explorer and Test Explorer tools are very handy. At the very least, every .NET developer should be spending time with the preview. ■

PETER VOGEL is a principal at PH&V Information Services, specializing in ASP.NET development with expertise in service-oriented architecture, XML, database and UI design.

THANKS to the following technical experts for reviewing this article:
Mike Abrahamson and Mike Fourie

DEVELOPED FOR INTUITIVE USE

DynamicPDF—Comprehensive PDF Solutions for .NET Developers

ceTe Software's DynamicPDF products provide real-time PDF generation, manipulation, conversion, printing, viewing, and much more. Providing the best of both worlds, the object models are extremely flexible but still supply the rich features you need as a developer. Reliable and efficient, the high-performance software is easy to learn and use. If you do encounter a question with any of our components, simply contact ceTe Software's readily available, industry-leading support team.



DynamicPDF

[WWW.DYNAMICPDF.COM](http://www.DynamicPDF.com)



TRY OUR PDF SOLUTIONS FREE TODAY!

www.DynamicPDF.com/eval or call 800.631.5006 | +1 410.772.8620

ceTesoftware

What's New for Web Development in Visual Studio 2012

Clark Sell

Do you remember the first Web site you developed? I sure do, and oh boy what a mess that was. But a great deal has changed since then—and it doesn't stop! It's getting harder and harder to remember what Web development was like a year ago, let alone when the Microsoft .NET Framework was first released.

Visual Studio has evolved as well. Remember when Web developers basically had to install a full server as a developer rig just to have IIS and SQL Server to run their sites? Then we were introduced to the more lightweight world of Cassini, and then IIS Express. Now HTML5 is here and Visual Studio embraces it.

I'm going to explore how Visual Studio supports this new-world Web developer, with a look at exciting new features such as project sharing, Page Inspector and DOM Explorer, as well as a look at how the core editing experience continues to evolve.

This article discusses a prerelease version of Visual Studio 2012. All related information is subject to change.

This article discusses:

- Project sharing
- Support for HTML5, JavaScript and CSS
- Debugging with Page Inspector
- The DOM Explorer

Technologies discussed:

Visual Studio 2012, HTML5, JavaScript, CSS

Project Sharing

You might notice the new Visual Studio features with the first click of your mouse, when you open a project file. In the past, project files were directly dependent on the editor you used. That meant if you had a project file created with Visual Studio 2010 and you tried to open it with Visual Studio 2012, you'd be asked to upgrade it. Once you did that, you could never again open the project in the older version. Well, not anymore.

The Visual Studio 2012 release candidate (RC) introduced the notion of project sharing. Now you can use it interchangeably with Visual Studio 2010 SP1 on the same source. Eliminating the requirement that everyone must use the same setup opens the door to new experiences, especially on diverse teams and open source projects—or even during software upgrades in your development shop.

Of course, there are a few caveats. For example, if your project uses a local database (the App_Data folder), you'll need to make sure you're still using the same version of the database engine. There



Figure 1 ARIA Roles



Figure 2 ARIA-* Properties



Figure 3 A Snippet for the Video Element



Figure 4 Intelligent Filtering and Indenting



Figure 5 The JavaScript Editor

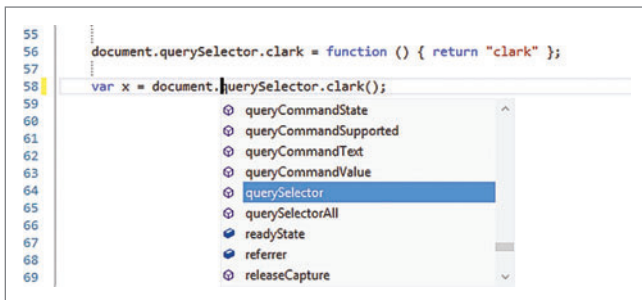


Figure 6 IntelliSense for JavaScript

are also a few older project types, such as ASP.NET MVC 2, that won't work with this.

As with anything new, you'll want to create a fork and do a few proofs of concept. The last thing you want to get caught doing is breaking the build process.

HTML5 Support

Probably the most significant feature to be added to the HTML editor in Visual Studio is Web Accessibility Initiative - Accessible Rich Internet Applications (WAI-ARIA) support. WAI-ARIA (w3.org/WAI/intro/aria) is a Web standard focused on making Web sites and Web applications more accessible to people with disabilities. The specification defines two types of attributes for your markup. Roles are applied to HTML elements to describe the type of widget

being used or the structure of the Web page. As you can see in **Figure 1**, Visual Studio support for ARIA roles includes IntelliSense.

The other type of attributes are prefixed with aria-, such as aria-busy or aria-valuemin, as shown in **Figure 2**. These aria-* attributes are used to help describe the element to a user.

HTML5 introduced more than 25 new semantic tags. Visual Studio already had IntelliSense support for these tags, but Visual Studio 2012 makes it faster and easier to write markup by adding some corresponding snippets, like the ones for the audio and video tags. **Figure 3** shows a snippet for the video tag. Though these tags aren't complicated, they come with a few small subtleties, such as adding the correct codec fallbacks.

The last few Web sites I built took advantage of the new HTML5 tags. With each of these sites, I ended up refactoring my markup in some way, even if it was just a simple semantic change, such as changing a div to a section. In Visual Studio 2012, when you change an element, the editor will also change its corresponding starting or ending element. This automatic renaming is another way the editor helps you avoid those late-night mistakes.

Developers often differ over whether to use tabs or spaces for formatting. No matter which you prefer, Smart Indent is here to keep you consistent. Regardless of your choice, after typing an element and hitting enter, you're taken to the next line, which is indented as you prefer. As you'd expect, IntelliSense has added filtering not only as you type but also based on the title case of the words in your searches, making it quicker and easier to insert elements. And when combined with Smart Indent, no matter where you drop the cursor, after hitting tab your new markup will automatically be positioned correctly in the document, as **Figure 4** shows.

Finally, the Visual Studio 2012 HTML editor has a number of great additions that support ASP.NET, such as Smart Tasks, extract user control and automatic event binding. You can find out more about these features on the "What's New in ASP.NET 4.5 and Visual Studio 2012" page at bit.ly/MXclbG.

JavaScript Support

The JavaScript editor in Visual Studio 2012 is completely new and supports ECMAScript 5. It brings some nice features like collapsing functions and brace matching. And if you're still scrolling around for a function, don't. Just hit F12 and get taken directly to the definition of that function or variable (see **Figure 5**).

Figure 7 A JavaScript Function That Uses the Signature Element

```
function myAwesomeFunction(a, b, c) {

    /// <signature>
    /// <summary>This is my awesome function</summary>
    /// <param name="a" type="String">Clearly you should pass A in here.</param>
    /// <param name="b" type="String">Clearly you should pass B in here.</param>
    /// <returns type="String" />
    /// </signature>
    /// <signature>
    /// <summary>This is my awesome function</summary>
    /// <param name="a" type="String">Clearly you should pass A in here.</param>
    /// <param name="b" type="String">Clearly you should pass B in here.</param>
    /// <param name="c" type="String">Clearly you should pass C in here.</param>
    /// <returns type="String" />
    /// </signature>

    return "yea pretty awesome";
}
```

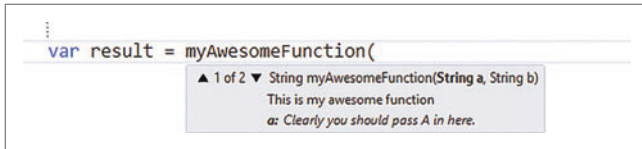



Figure 8 Comments Added to IntelliSense

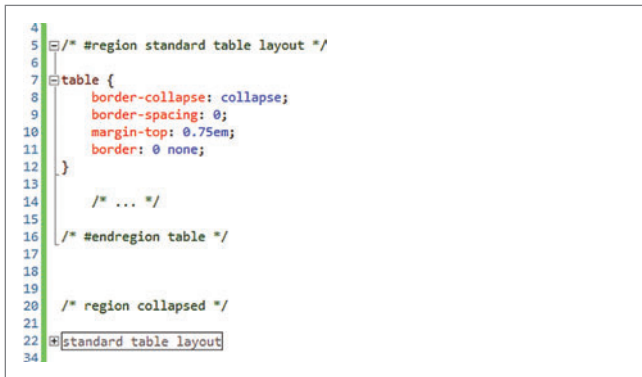


Figure 9 Regions

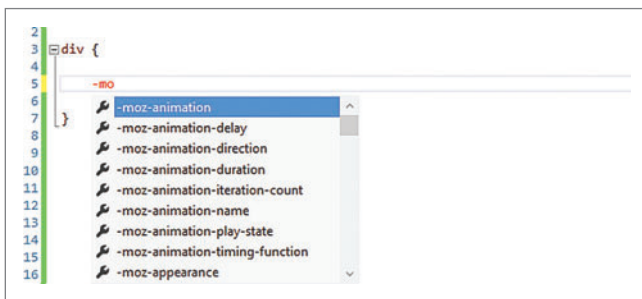


Figure 10 Properties for Vendor Prefix -moz



Figure 11 CSS IntelliSense

The editor also enhances IntelliSense for JavaScript development and, along with it, improved support for the Document Object Model (DOM). With HTML5 becoming mainstream, it's great to have this enhanced support not only for the basic new DOM APIs but also when you decide to extend them, as **Figure 6** shows.

Documentation is always a good thing to have, especially when it's the kind of documentation that lives in the IntelliSense window. VSDoc support is not new to Visual Studio, but it now includes a new signature element for declaring overloads of JavaScript functions that let you create detailed IntelliSense comments. Ignoring my code quality for a second, take a look at **Figure 7**, which shows a function that takes either two or three arguments. As you can see, the code has a series of <signatures> for each overload, with the results shown in **Figure 8**.

With that simple documentation in place you can see in **Figure 8** how IntelliSense used it for the contents displayed.

CSS3 Support

For me, CSS continues to be a mystical art, and one I can never get enough help with. Luckily, Visual Studio 2012 has enhanced support for CSS. Regions, IntelliSense and snippets are just a few features that can help improve your style. Let's start by looking at regions. All it takes is a special comment to create a region. **Figure 9** shows a simple region both expanded and collapsed for styling a table.

As you know, in the Web world you must support multiple browsers. With CSS, that means accounting for all of the different vendor prefixes. The Visual Studio 2012 CSS editor now includes vendor prefixes in both the standard list of properties and also in the built-in snippets, as shown in **Figure 10**.

Of course, there are many CSS properties at your disposal and filtering is essential. As with IntelliSense for C# and Visual Basic, you can filter that long list of CSS properties just by typing, as **Figure 11** shows in the search for `borr` (border-radius).

This is great, but for border-radius you should really support all of the vendor prefixes. If you look closely at the border-radius icon, you'll see it's actually a snippet. Hit the tab twice, once to auto-complete the property and again to insert the snippet (see **Figure 12**).

You can see that all of the vendor prefixes were added and the value is now highlighted for you to change. As you type the desired value for the radius, all vendor-specific values will change accordingly. But what about something more advanced, like a media query? Type `@` and you'll see a list of advanced snippets such as `@media`. Select `@media` then hit tab and the media query snippet will be inserted. Now you can just type in your new width, hit tab and adjust your height (see **Figure 13**).

Snippets are a fantastic way to save on keystrokes. Just as in C# or Visual Basic, you can create new snippets or modify existing ones

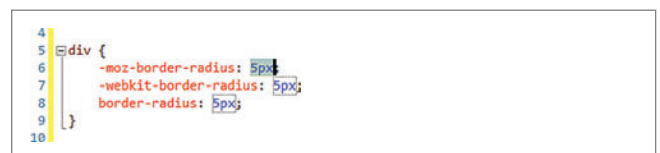


Figure 12 A CSS Snippet



Figure 13 The @media Snippet

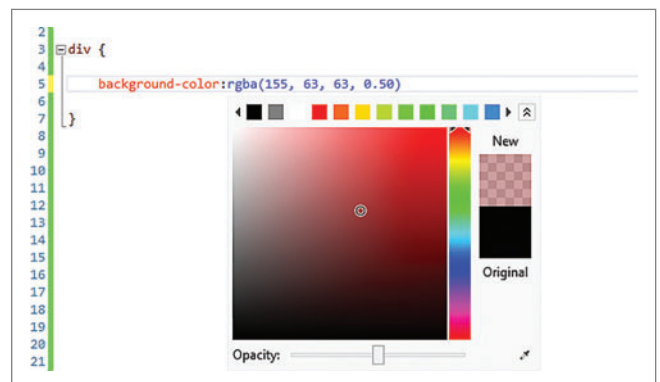


Figure 14 The New CSS Color Picker

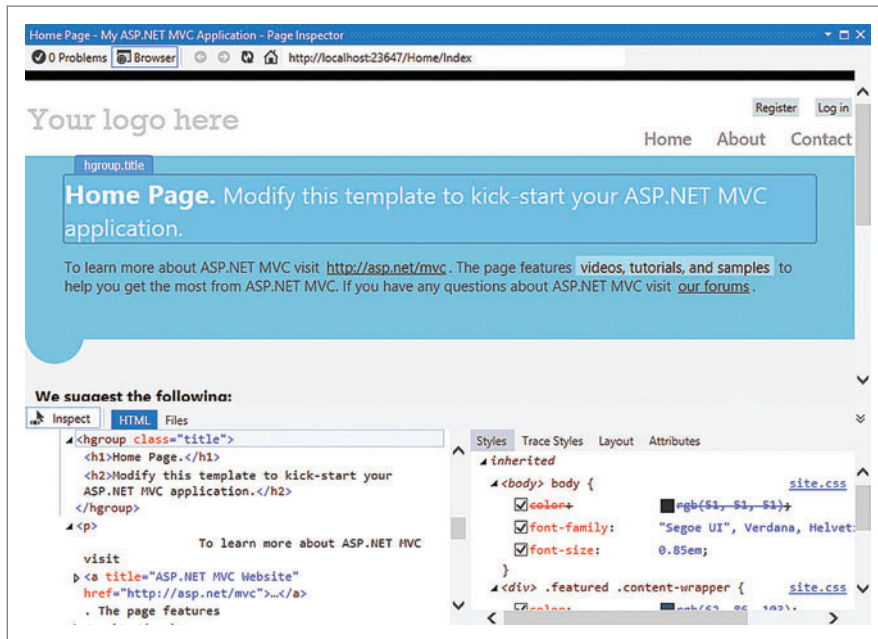


Figure 15 Page Inspector

through the Visual Studio Code Snippet Manager, which you'll find at Tools | Code Snippet Manager.

Visual Studio also has a new color picker for those properties that require color; it replaces the named colors in previous versions. As you can see in Figure 14, I've chosen to pick a color with a bit of transparency, so the color picker changed from a standard hex color to rgba.

Finally, be sure to check out the support for hierarchical indentation and CSS hacks at bit.ly/MXclbG.

Debugging with Page Inspector

"Hey, it works on my machine!" I guarantee you've either said that phrase or heard it more than once during your career. In Web development, it's almost guaranteed you'll hear that phrase sooner or later, and why is that? Well, in part it's because what we deploy to and what we develop on are not identical.

Leaving aside the obvious things like security and server farms, it really comes down to what the browser will execute and render. Over the years, technologies such as HTML5, CSS3 and JavaScript have advanced, and we use a vast array of frameworks, including some that might only ever run in the browser (such as knockout.js and backbone.js). Web developers have to deal with two different worlds: how the app is developed and how it runs across all of the different browsers that need to be supported.

Browser tools have advanced accordingly, which helps a great deal, but you still might need an archaeological dig to find the original source file. Even if you

consider simple JavaScript minification and script loaders, what might be sent to the server could be drastically different from what you've developed. Thankfully, Page Inspector, one of my favorite additions to Visual Studio 2012, is here to help. Page Inspector brings browser diagnostics tools directly into Visual Studio. By doing so, it provides an integrated experience from the browser to ASP.NET and right to the source code. It does all of this with minimal setup.

You can run Page Inspector out of the box with limited functionality. To do so, just right-click on the page you want to view and select View in Page Inspector. Now, I know what you're thinking: "There's no way that will work on my MVC project or with all of my custom routes." Have some faith.

Page Inspector will do its best to actually map the file to the URL convention used. Where it fails, you can manually add that

mapping. To fully enable Page Inspector and all of its features, just add a new key to the appSettings section of your web.config:

```
<add key="VisualStudioDesignTime:Enabled" value="true" />
```

Page Inspector is all about plugging that "gap" I mentioned earlier, between what was developed and what was rendered. Wouldn't it be nice if you could just highlight something in the rendered output and find what file it's in—or, better yet, change it? I know you can do that already in the browser tools, but I mean for good, actually in the source it came from. With Page Inspector you can do this, as it maps between the rendered output and the source files it came from. When you start Page Inspector, you're basically presented with a browser window inside Visual Studio, as shown in Figure 15.

There's a lot going on here, but let's break it down:

- The chrome of the window is a browser frame. You can see the address bar at the top.
- The top half of the page is the actual Web page as rendered to the browser.

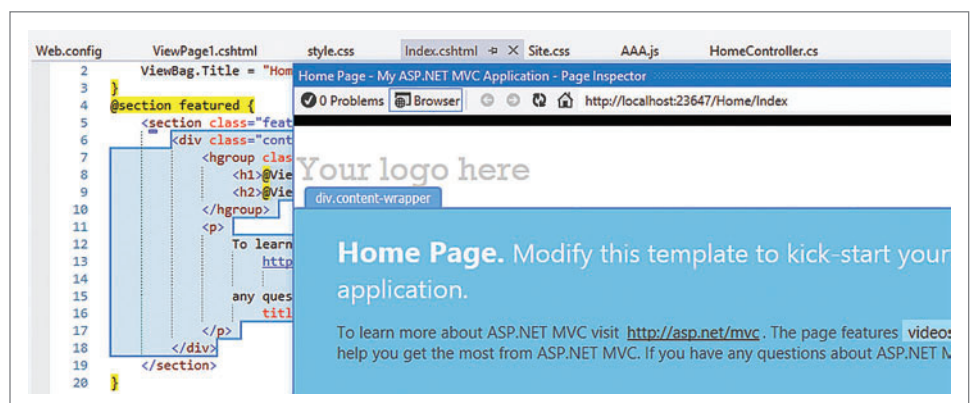


Figure 16 Mapping Between Source and Rendered Output

- The bottom left is the HTML markup for the rendered page. This is because HTML was selected in the tab directly above.
- The bottom right is the style for the rendered page.

Similar to the tools found in the browser, there's a feature called Inspect. When you select Inspect, as you move around the page you'll see the different DOM elements highlighted and labeled. In **Figure 15**, the element is hgroup with a class called title. You'll also see in the HTML window the corresponding markup highlighted. This happens in real time as you move around the document. As if that weren't cool enough, take a look at **Figure 16**. I've selected or hovered over an element in Page Inspector and behind it you can see it's highlighted in the correct source file where that text resides. This doesn't just happen at the element level but all the way down to character. It's a live character-by-character, two-way mapping from the source to the rendered output. Regardless of what you select (source or rendered output), you'll see it highlighted.

I've just scratched the surface of the capabilities of Page Inspector. This is one of those features that you need to use to fully appreciate how it can make your life easier. Prepare to be amazed. Next time you're in a Web project, right-click on a page and select View in Page Inspector.

Publishing

What good is a Web site if you can't deploy it? Visual Studio has long had support for deploying sites, including a feature called Publish Profile. Historically, once a profile was created it was only a local machine asset. In Visual Studio 2012, this profile is now part of the overall project assets. It's a simple MSBuild file, which will get imported into your overall MSBuild chain. Here's what a simple FTP profile might look like:

```
<Project ToolsVersion="4.0"
  xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <WebPublishMethod>FTP</WebPublishMethod>
    <SiteUrlToLaunchAfterPublish>http://thatconference.com
    </SiteUrlToLaunchAfterPublish>
    <publishUrl>ftp://thatConference.com</publishUrl>
    <DeleteExistingFiles>True</DeleteExistingFiles>
    <FtpPassiveMode>True</FtpPassiveMode>
    <UserName>foooooo</UserName>
    <_SavePWD>True</_SavePWD>
  </PropertyGroup>
</Project>
```

You can find these profiles saved in the overall project properties folders:

- C#—Properties\PublishProfiles
- Visual Basic—MyProject\PublishProfiles

Because it's an MSBuild project file, of course you can call it from the command line, like so:

```
msbuild.exe myProject.csproj /t:WebPublish /p:PublishProfile=ProfileName
```

HTML5 on Windows 8

At last year's BUILD conference, Microsoft announced a new programming model for building native apps on Windows 8. As it turned out, this new programming model isn't really new at all; part of the development story is simply a new reliance on HTML5, CSS3 and JavaScript to build applications. This means anyone with a Web development background has a skill set applicable for writing native apps for Windows 8. Of course, Visual Studio is right there to help.

Snippets are a fantastic way to save on keystrokes.

A number of the core editing improvements I've discussed here are still applicable for developing Windows Store apps, with the exception of Page Inspector (because Windows Store apps don't run in a browser). How, then, do you debug HTML5 Windows Store applications? With the DOM Explorer and the JavaScript Console, as shown in **Figure 17**.

As with Page Inspector, you can select elements from the running application and go directly to the location in the source. You can see the styles applied and change values while the application is running. Want to put a breakpoint in some JavaScript? Go ahead, and then do some live debugging. You can even fire up the JavaScript Console and start hacking around. Regardless of whether you're building a Web site or an application for Windows 8, it's the same, with the same features across both platforms in one great editor.

If you're reading this, chances are pretty good that you use Visual Studio. I've used it myself for more than a decade and I've learned with every major release that it's good to remind yourself to explore all of the new features. It's too easy sometimes to just stick with what

you already know, so go and poke around. It's great to see Visual Studio embrace current technologies like HTML5, CSS3 and ECMAScript 5, while putting facilities in place for upgrading as standards change. ■

CLARK SELL works as a senior Web and Windows 8 evangelist for Microsoft outside of Chicago. He blogs at csell.net, podcasts at DeveloperSmackdown.com and can be found on Twitter at twitter.com/csell5.

THANKS to the following technical experts for reviewing this article:
Matt Carter and Orville McDonald

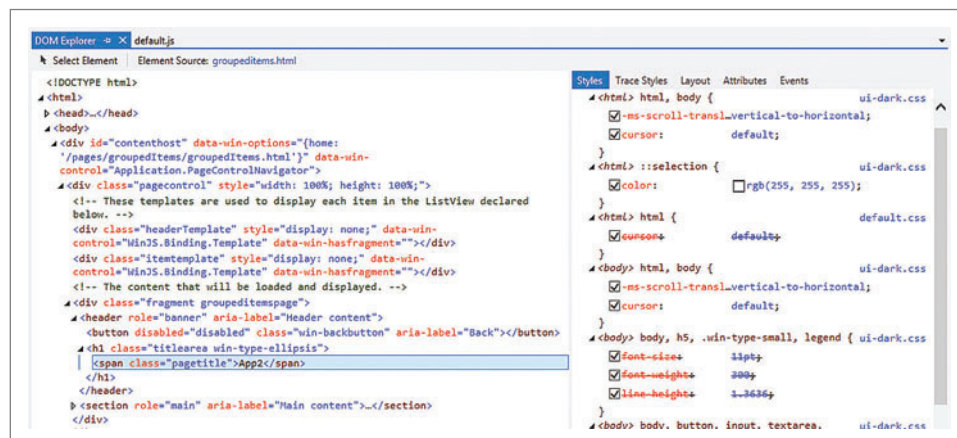


Figure 17 The DOM Explorer

Does your Team do more than just track bugs?

Free Trial and Single User FreePack™ available at www.alexcorp.com

Alexsys Team® does! Alexsys Team 2 is a multi-user Team management system that provides a powerful yet easy way to manage all the members of your team and their tasks - including defect tracking. Use Team right out of the box or tailor it to your needs.



Alexsys Team

Track all your project tasks in one database so you can work together to get projects done.

- Quality Control / Compliance Tracking
- Project Management
- End User Accessible Service Desk Portal
- Bugs and Features
- Action Items
- Sales and Marketing
- Help Desk

Native Smart Card Login Support including Government and DOD



New in Team 2.11

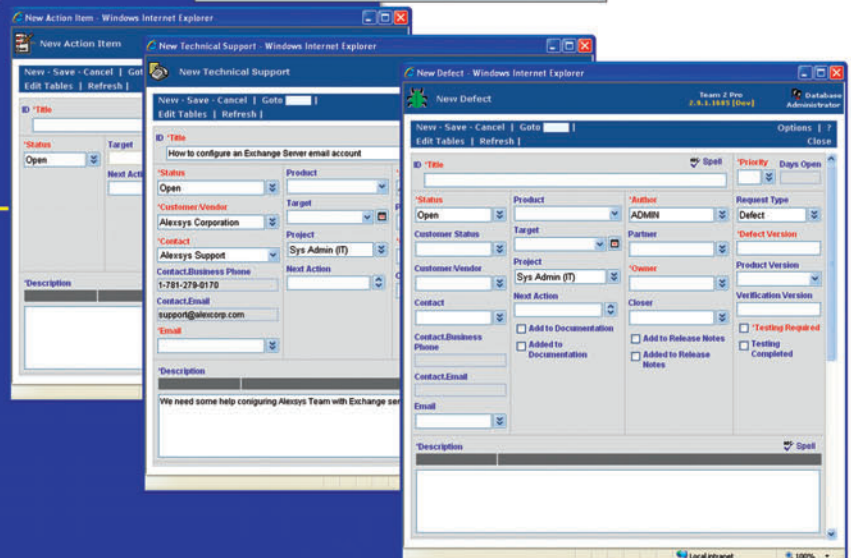
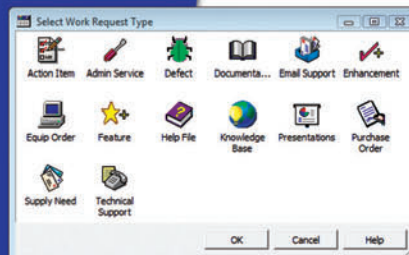
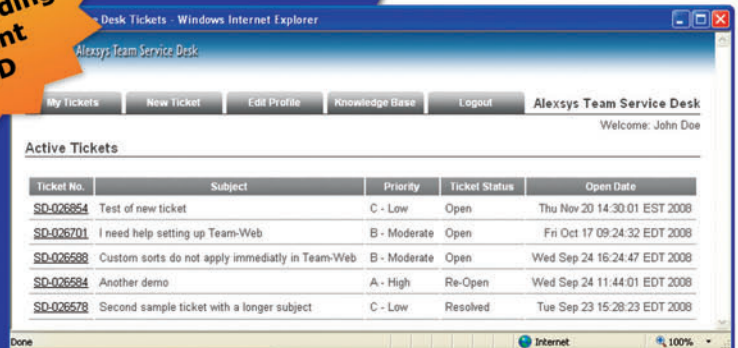
- Full Windows 7 Support
- Windows Single Sign-on
- System Audit Log
- Trend Analysis
- Alternate Display Fields for Data Normalization
- Lookup Table Filters
- XML Export
- Network Optimized for Enterprise Deployment

Service Desk Features

- Fully Secure
- Unlimited Users Self Registered or Active Directory
- Integrated into Your Web Site
- Fast/AJAX Dynamic Content
- Unlimited Service Desks
- Visual Service Desk Builder

Team 2 Features

- Windows and Web Clients
- Multiple Work Request Forms
- Customizable Database
- Point and Click Workflows
- Role Based Security
- Clear Text Database
- Project Trees
- Time Recording
- Notifications and Escalations
- Outlook Integration



Free Trial and Single User FreePack™ available at www.alexcorp.com. FreePack™ includes a free single user Team Pro and Team-Web license. Need more help? Give us a call at 1-888-880-ALEX (2539).

Team 2 works with its own standard database, while Team Pro works with Microsoft SQL, MySQL, and Oracle Servers.
Team 2 works with Windows 7/2008/2003/Vista/XP.
Team-Web works with Internet Explorer, Firefox, Netscape, Safari, and Chrome.

Developing and Deploying Windows Azure Cloud Services Using Visual Studio

Boris Scholl and Paul Yuknewicz

The Windows Azure SDK for .NET has come a long way since its first release. Functionality has been regularly added as the Windows Azure platform and associated tools evolved, such as adding Windows Azure Cloud Service projects to any Web app project, ASP.NET MVC versions 3 and 4 project support, and a dramatically streamlined publishing experience. This aligns with Microsoft's overall vision to improve the tools used for cloud development and to better integrate with all aspects of the application development lifecycle.

As of June 2012, Windows Azure offers three compute container options to develop and run applications. These options include

Windows Azure Web Sites (Preview) for quick and easy Web site and Web application deployment; Windows Azure Virtual Machines (Preview) for durable Windows Server and Linux Infrastructure as a Service (IaaS) virtual machines (VMs) and applications running on them; and Windows Azure Cloud Services, which provide reserved, infinitely scalable, *n*-tier options running on a Platform as a Service (PaaS). This article focuses on developing cloud services; you can learn more about all of the options at windowsazure.com.

We'll walk through parts of the cloud service application development lifecycle using Visual Studio and highlight the new features as we progress through that lifecycle. After reading through the article, readers new to Windows Azure should have a basic understanding of cloud development with Visual Studio, and readers who already have experience with Windows Azure development in Visual Studio should have a good understanding of the new features available.

Getting Started

The Windows Azure SDK for .NET June 2012 release includes Visual Studio tools, which work on top of Visual Studio 2010 SP1 and Visual Studio 2012 release candidate (RC) and above. For this article, we'll use Visual Studio 2012 RC. The best way to install the tools is to start Visual Studio, open the New Project dialog box and select the Cloud node. This will show the Get Windows Azure SDK for .NET project template.

That link will take you to the .NET Developer Center (bit.ly/v5MF7m) for Windows Azure and highlight the all-in-one installer for the SDK.

This article discusses previews of Windows Azure components and a prerelease version of Visual Studio 2012. All related information is subject to change.

This article discusses:

- Creating Windows Azure projects
- Converting to cloud service projects
- Debugging locally
- Publishing to Windows Azure
- Working with Server Explorer
- Troubleshooting in Visual Studio

Technologies discussed:

Windows Azure SDK for .NET - June 2012, Visual Studio 2010 SP1, Visual Studio 2012



IT EVENTS WITH PERSPECTIVE

What is Live! 360?

Brought to you by the publishers of *MSDN Magazine*, Live! 360 is a new IT conference comprised of four co-located and technology-based events. Get leading-edge training, unique networking opportunities and a chance to preview the future of technology – all in one conference.

Save Up To \$400!

**Register Before
October 10**

Use Promo Code TIP

live360events.com

Buy 1 Event, Get 3 Free!

Customize an agenda to suit your needs – attend just 1 event or all 4 for the same price.

Visual Studio 
EXPERT SOLUTIONS FOR .NET DEVELOPERS

SharePoint 
TRAINING FOR COLLABORATION

SQL Server 
TRAINING FOR DBAS AND IT PROS

Cloud & Virtualization 
THE FUTURE OF COMPUTING



Orlando, FL December 10-14

Royal Pacific Resort at Universal Orlando | live360events.com

More event details on the back! ►



Visual Studio

EXPERT SOLUTIONS FOR .NET DEVELOPERS

Code in the Sunshine!

Developers, software architects, programmers and designers will receive hard-hitting and practical .NET Developer training from industry experts and Microsoft insiders. **vslive.com/orlando**

SharePoint

TRAINING FOR COLLABORATION

Build. Develop. Implement. Manage.

Leading-edge knowledge and training for SharePoint administrators, developers, and planners who must customize, deploy and maintain SharePoint Server and SharePoint Foundation to maximize the business value. **splive360.com**

SQL Server

TRAINING FOR DBAs AND IT PROS

Bringing SQL Server to Your World

IT Professionals, DBAs and Developers – across a breadth of experience and organizations – come together for comprehensive education and knowledge share for SQL Server database management, performance tuning and troubleshooting. **sqllive360.com**

Cloud & Virtualization

THE FUTURE OF COMPUTING

Cloud and Virtualization for the Real World

The event for IT Professionals, Systems Administrators, Developers and Consultants to develop their skill sets in evaluating, deploying and optimizing cloud and virtual-based environments. **virtlive360.com**

live360events.com

GOLD SPONSOR



SUPPORTED BY



PRODUCED BY



MEDIA SPONSOR



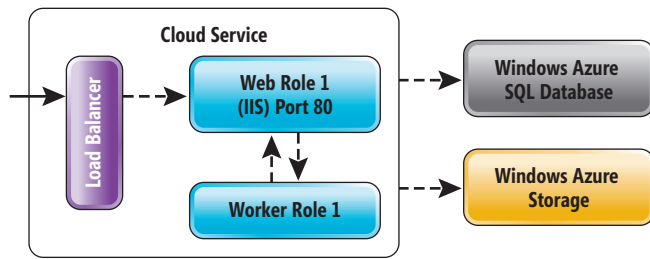


Figure 1 Common Components of a Cloud Service

Creating Windows Azure Projects

Once everything is installed, you can go ahead and create a Windows Azure Cloud Service project, the focal project for working with Windows Azure Cloud Services. The cloud service is a compute container for infinitely scalable, highly available and multitier cloud applications. A helpful new feature in this SDK release is that it works side by side with the Windows Azure SDK for .NET November 2011 (1.6) release. This means you can still work on your 1.6 projects without having to upgrade them. In general, you have two options for creating a Windows Azure application. The first approach is to create a Windows Azure project from scratch. To do that, start Visual Studio with elevated privileges, click on the File menu and choose New | Project to bring up the New Project dialog. Under Installed Templates | Visual C# (or Visual Basic or F#), select the Cloud node and select the Windows Azure Cloud Service Project template to bring up the New Cloud Service Project dialog. This dialog lets you add roles to a cloud service.

If the June 2012 SDK is installed side by side with the November 2012 SDK, a dropdown for the SDK version is shown at the top the dialog, which enables you to choose the SDK version with which to create the role.

Before we proceed, let's quickly review what roles and instances mean. A role is basically a definition for both the app and the PaaS-managed VM it will run on, defining, for example, which OS and modules should be installed on the VM, which diagnostics settings should be used and what endpoints are supposed to be exposed. Think of the role as your template that lets you stamp out as many or as few instances (that is, VMs) as you need to scale to the current demands on your cloud service. There are currently two types of roles you can create from Visual Studio:

1. **Web Role:** A Web application running on IIS. It's accessible via an HTTP or HTTPS endpoint. Typically this is used for front-end Web applications and Web services.
2. **Worker Role:** A background processing application that runs arbitrary .NET code. It also has the ability to expose Internet-facing and internal endpoints on HTTP, HTTPS, TCP and User Datagram Protocol.

Instances directly correspond to VMs in the cloud running roles defined by the role template.

Figure 1 shows the components of a cloud application and how they work together.

Figure 2 Initializing the Diagnostics Monitor and Adding Some Windows Event Log Data Sources

```
public class WebRole : RoleEntryPoint
{
    public override bool OnStart()
    {
        DiagnosticMonitorConfiguration diagConfig =
            DiagnosticMonitor.GetDefaultInitialConfiguration();
        diagConfig.WindowsEventLog.DataSources.Add("System!*");
        diagConfig.WindowsEventLog.DataSources.Add("Application!*");
        diagConfig.WindowsEventLog.ScheduledTransferPeriod =
            TimeSpan.FromMinutes(1.0);
        diagConfig.WindowsEventLog.ScheduledTransferLogLevelFilter = LogLevel.Error;

        DiagnosticMonitor.Start(
            "Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString", diagConfig);

        return base.OnStart();
    }
}
```

As noted, another compute container type introduced with the June 2012 release is the Windows Azure Virtual Machine (Preview). VMs allow you to create completely custom workloads with any OS or any software (for example, databases, app servers or legacy components). This is great to add a custom tier, which can be used in conjunction with cloud services. You can use this container to create VMs from scratch or use VMs provided by the VM gallery in Windows Azure. At the time this article was written, you need to first create VMs in the management portal (rather than from within Visual Studio), but as you'll see later, you can explore VMs and their properties in Server Explorer, so it's easy to reference VM workloads in your cloud service code. Note that unlike Web and Worker Role VMs that are managed and updated by Microsoft, you'll be on your own to update and manage your Windows Azure Virtual Machines, which can be used in conjunction with cloud services.

If your application is under heavy load, you might need to scale out. You can do that by adding more instances to your cloud service.

After adding roles to the cloud service and clicking OK, Visual Studio will create a solution that includes the cloud service project and a project corresponding to each role you added.

The Web Roles are ASP.NET Web application projects with only a few differences. The Web Role project, MVCWebRole1, contains references to the following libraries that are not referenced with a standard ASP.NET Web application:

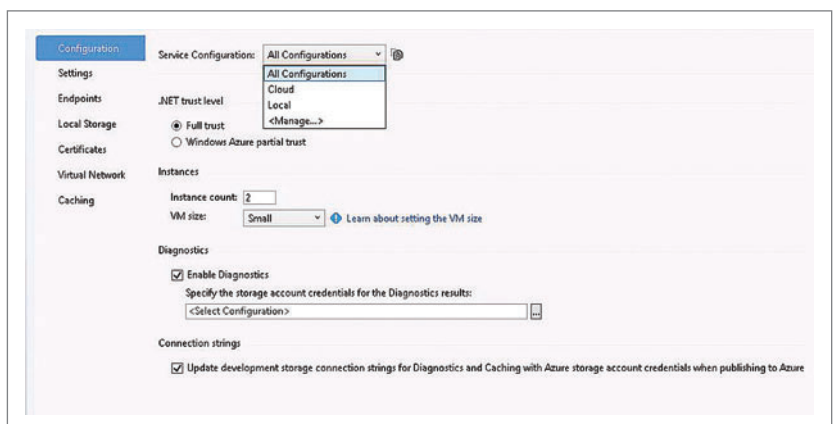


Figure 3 Service Configuration Files

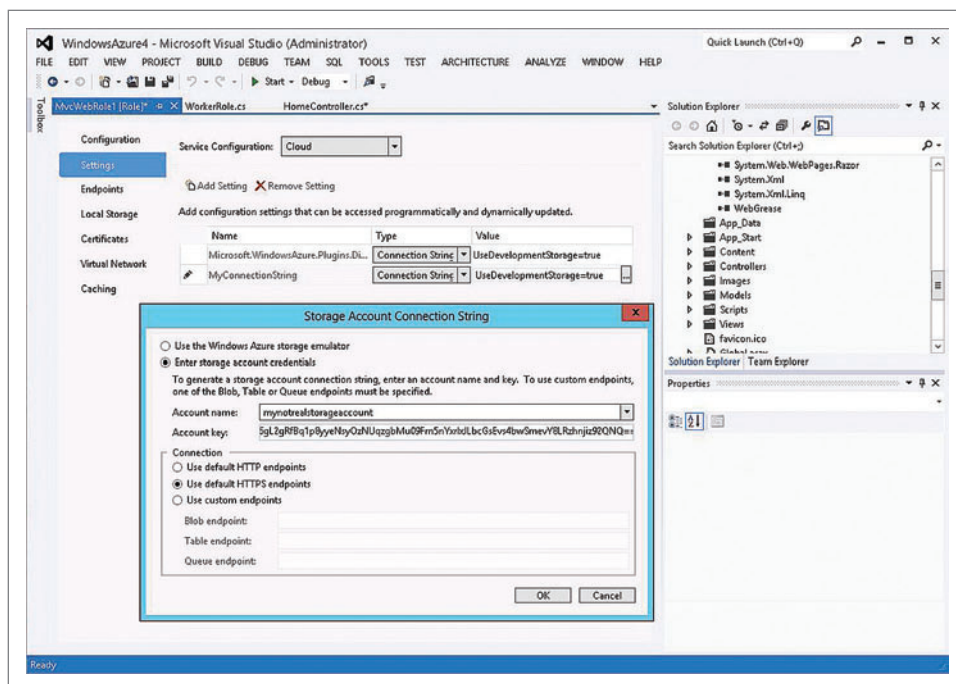


Figure 4 Storage Account Connection String Builder for Cloud Configuration

- **Microsoft.WindowsAzure.Configuration:** This is a configuration helper library. This assembly is new in the June 2012 release and enables developers to get configuration values of Windows Azure services regardless of whether the application is hosted on-premises, in a cloud service, in Windows Azure Web Sites or in an IaaS VM. That means regardless of which configuration file (web.config or cloud.cscfg) contains your connection string value or any other setting, you can access the value by using the same method. The following sample shows how to use the CloudConfigurationManager to read a key "MyConnectionString" from a configuration file:

```
private string conn = CloudConfigurationManager.GetSetting(
    "MyConnectionString");
```

- **Microsoft.WindowsAzure.Diagnostics:** This contains the diagnostics and logging APIs to enable and configure diagnostics on Windows Azure for your application. Figure 2 shows how to initialize the diagnostics monitor and add some Windows event log data sources in the OnStart method of a role in WebRole.cs.
- **Microsoft.WindowsAzure.ServiceRuntime:** This includes environment and runtime APIs for Windows Azure. The following sample shows how to use the CurrentRoleInstance property of the RoleEnvironment class:

```
var roleInstance = RoleEnvironment.CurrentRoleInstance;

• Microsoft.WindowsAzure.StorageClient: This is the .NET API to access the Windows Azure storage services for Binary Large Objects (BLOBs), tables and queues. The following code shows how to create a storage account object and create a new client for a BLOB service:
```

```
CloudStorageAccount storageAccount =
    CloudStorageAccount.Parse(CloudConfigurationManager.GetSetting(
        "MyConnectionString"));
CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();
```

With the exception of Microsoft.WindowsAzure.Diagnostics and Microsoft.WindowsAzure.ServiceRuntime, all the preceding assemblies mentioned are referenced as NuGet packages, which makes it easier to service your application when newer versions are available.

The cloud service project contains the roles that are included in the cloud service, along with the definition and configuration files. It provides Windows Azure-specific run, debug and publish functionality.

You can use the New Project | Cloud | Windows Azure Cloud Service dialog to create a cloud service with any number of Web and Worker Roles and use a different template for each role. Once the project is created, you need to configure each role by double-clicking on the role node—in

our example, MvcWebRole1 under the Roles folder—in Solution Explorer. The role designer allows you to configure important role-related settings. For example, you can set the number of instances of each role independently to improve performance and provide redundancy. In fact, Windows Azure requires at least two instances in order to ensure the high availability defined in the compute service-level agreement.

All the settings are stored in the Windows Azure Service configuration file. Starting with the November 2011 1.6 release, you can create multiple configuration files to support different scenarios. For example, you might want to have a service configuration that contains only two instances for your staging environment and four instances in your production environment. By default, there

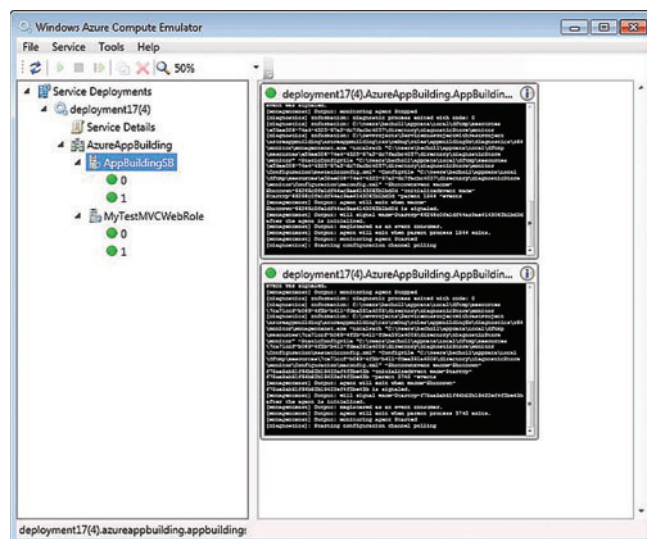


Figure 5 The Windows Azure Compute Emulator

Visual Studio LIVE!

EXPERT SOLUTIONS FOR .NET DEVELOPERS



Orlando, FL December 10-14

Royal Pacific Resort at Universal Orlando | vslive.com/orlando

Who Should Attend?

- Developers
- Software Architects
- Programmers
- Designers
- IT Professionals
- IT Managers
- Consultants

Session Topics Include:

- HTML5
- Mobile
- Visual Studio / .NET
- Windows 8 / WinRT
- WPF / Silverlight
- XAML

Save Up To \$400!

**Register Before
October 10**

Use Promo Code DEVSS

vslive.com/orlando



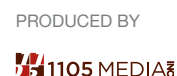
Buy 1 Event, Get 3 Free! Customize an agenda to suit your needs – access to all 4 Live! events is included in your registration. Attend just 1 conference or all 4 for the same price.

Visual Studio Live! is co-located with:

Cloud & Virtualization LIVE!
THE FUTURE OF COMPUTING

SQL Server LIVE!
TRAINING FOR DBAS AND IT PROS

SharePoint LIVE!
TRAINING FOR COLLABORATION



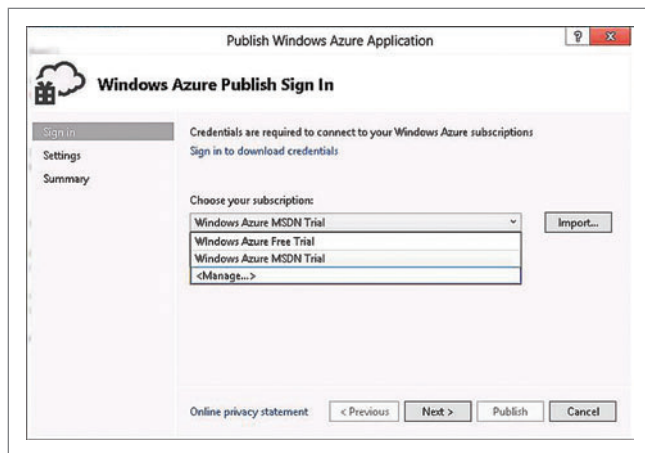


Figure 6 The Publishing Wizard

are configuration files for local and cloud environments, but you can also create custom configs using the <Manage...> entry in the Service Configuration dropdown, as shown in **Figure 3**. The Windows Azure management portal provides support for changing some of the configuration settings using a Web UI post-deployment.

To best explain how that works, you can look at how we deal with storage connection strings. Let's assume you want to build an application that stores customer data such as names and phone numbers. When running locally—for example, in debugging mode—the data will be stored in the local storage used by the storage emulator or a local database engine such as SQL Server 2012 LocalDB. Local storage configuration won't work in the cloud because there's no persistent storage engine in your role like LocalDB, so instead you want to take advantage of the Windows Azure storage options such as Windows Azure Storage or Windows Azure SQL Database. The connection string also points to local storage or the localhost DNS name, so that won't work in the cloud.

Instead, you need to specify a Windows Azure storage account in the connection string to match the target environment in Windows Azure. This is where the multiple configurations feature comes in handy. You can define settings such as connection strings in the Settings section of the role designer. Here you can add a new setting of type Connection string and call it, for example, "MyConnectionString."

You can now enter the connection string by clicking the ellipsis button to bring up the Storage Account Connection String builder. First, you add the value for your local storage by selecting the option "Use the Windows Azure storage emulator."

Next, you enter the connection string you want to use for your published app. To do that, you need to switch the service configuration setting (shown in **Figure 3**) to Cloud and bring up the Storage Account Connection String builder again, as shown in **Figure 4**.

You can now enter the storage account credentials you want to use to store your customer data. The account key can be obtained by using

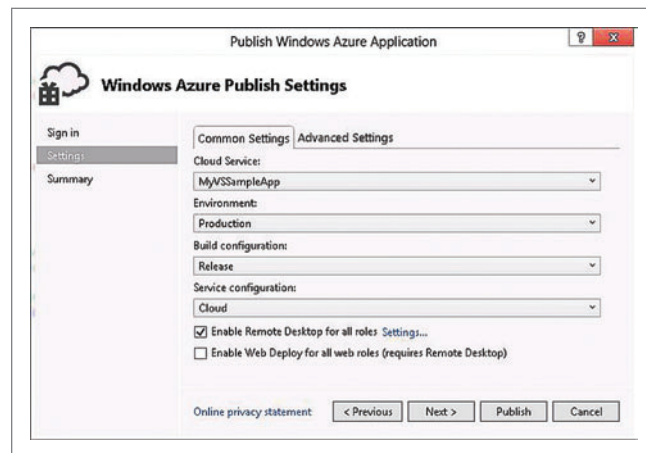


Figure 7 Common Settings Tab

the "manage keys" option in the storage section of the Windows Azure Management Portal.

As mentioned, you can now access the connection string value in your code by using the CloudConfigurationManager object like this:

```
var conn = CloudConfigurationManager.GetSetting("MyConnectionString");
```

Visual Studio defaults to using the local configuration for F5, so your app will automatically use local storage for debugging. When you publish with the Cloud configuration selected (default setting), your code will use the connection string pointing to Windows Azure instead. You'll see later how to choose the service configuration when you're about to publish the application to Windows Azure.

As you can see, one big benefit of supporting multiple configuration files is that you don't need to update the Windows Azure service configuration each time you publish to a different target environment, nor is there a need to maintain multiple Windows Azure cloud service projects.

Another interesting new role aspect of the June 2012 release is the Windows Azure Caching (Preview). Like any caching solution, the aim is to improve performance and latency. Unlike other solutions, this one runs in your role instances and can serve as a distributed and highly available service to your role instances. It can be used as a provider for existing caching APIs such as ASP.NET caching, output cache, session-state cache or memcache. Caching is enabled and configured in a role. You can simply check the Enable Caching checkbox to get smart defaults and get caching up and running immediately.

Existing roles with excess memory can be co-located with the cache service, or you have the option to create a new dedicated cache role. The co-located option has the benefit that you can take advantage of the excess resources in role instances for which you're already paying. Each in-role cache service can be configured to host multiple named caches that have individual policies that control each named cache's availability, eviction policy and more. Each named cache can be configured for memcache compatibility and accessed programmatically,

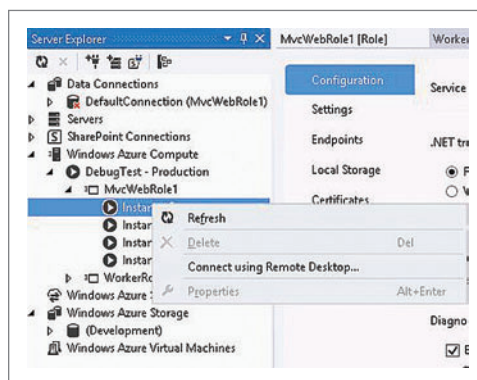


Figure 8 Connect Using Remote Desktop

SharePoint® **LIVE!**

TRAINING FOR COLLABORATION



Orlando, FL December 10-14

Royal Pacific Resort at Universal Orlando | splive360.com

SharePoint Live! provides training for those who must customize, deploy and maintain SharePoint Server and SharePoint Foundation to maximize the business value. Both current & newly released versions will be covered!



Who Should Attend?

- SharePoint Administrators
- Power Business Users
- Managers
- IT Pros
- Developers
- Project Managers
- Information Architects

Session Topics Include:

- Strategy, Governance, Adoption
- Management and Administration
- Information & Content Management
- BI, BPA, Search, and Social
- SharePoint and the Cloud
- SharePoint 2010 & SharePoint 2013

Save Up To \$400!

**Register Before
October 10**

Use Promo Code DEVSPS
splive360.com



Buy 1 Event, Get 3 Free! Customize an agenda to suit your needs – access to all 4 Live! events is included in your registration. Attend just 1 conference or all 4 for the same price.

SharePoint Live! is Co-located with:



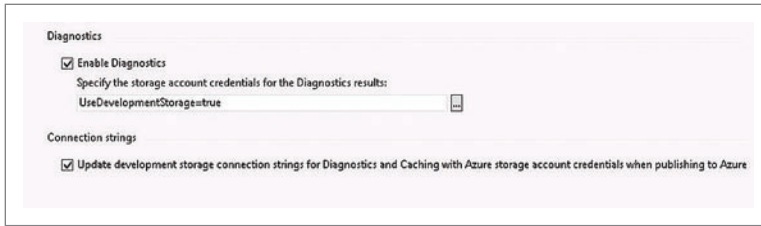


Figure 9 Update Development Storage Connection Strings for Diagnostics and Caching

and serve as the backing store for existing caching solutions such as output cache or session-state cache. The new in-role cache service is compatible with the on-premises ApplicationServer APIs for caching to allow easy migration from on-premises to the cloud. As mentioned, you can even create a dedicated Caching Role by choosing the Cache Worker Role template when adding a new role to your project. More information on how to set up caching can be found in the MSDN Library article, “How To: Use Windows Azure Caching (Preview),” at bit.ly/LRFStZ.

Turning Web Application Projects into Cloud Service Projects

In addition to the traditional way of creating a Windows Azure project, you can also “Azurify” an existing Web Application project. Let’s assume you already have a Web Application project such as an ASP.NET MVC 4 project that you want to deploy to Windows Azure. Right-click on your project and choose the “Add Windows Azure Cloud Service Project” command.

This adds a cloud service project to the solution. Visual Studio also adds the NuGet references needed for a Windows Azure project to the MVC project. In addition, the tool sets the Copy Local property for the System.Web.Mvc assembly to *true* because the assembly isn’t available on Windows Azure.

Debugging a Windows Azure Application Locally

Once you’ve created your project, you’ll likely need to debug it. You should always remember that Visual Studio needs to be started with admin-elevated privileges in order to debug a Windows Azure application. As with any other Visual Studio project, you can start debugging by setting a breakpoint and hitting F5. It’s worth mentioning that in the 1.7 SDK release, Visual Studio and the Compute Emulator use IIS Express to host instances and LocalDB for development storage by default, as opposed to IIS and SQL Express in the 1.6 SDK release (don’t worry, these options still exist in the project properties).

When debugging, Visual Studio will automatically use the local service configuration. As with previous versions, you can use the Windows Azure Compute Emulator to perform various operations on your deployments, such as viewing logs and restarting and deleting deployments (see **Figure 5**). To bring up the Compute Emulator, right-click on the Windows Azure notification icon in the taskbar and click on Show Compute Emulator UI. Notice that the Windows Azure Compute Emulator contains a new deployment that hosts two Web Role instances and two Worker Role instances.

Publishing to Windows Azure

Now that you’ve created, edited and debugged your application locally, you’re ready to deploy it to Windows Azure. In general, it’s good practice to follow the application development lifecycle before doing a final publish to the Windows Azure production environment. First, you should publish the application to a test environment. The test environment is basically a cloud service that you need to create and use for testing purposes only. This environment allows you to test if the application behaves as expected when hosted in Windows Azure. Once the tests are successful, you can publish to the staging environment. In this environment, you can do user-acceptance tests to validate if the application provides the functionality for which it was designed. Finally, if all the tests pass, you can publish to the production environment.

The publishing process was improved considerably in the November 2011 release with the introduction of a new publishing wizard. To open the wizard, right-click on the Windows Azure project and select Publish. Note that you need to right-click on the Windows Azure project, not on a Web Application project such as an ASP.NET MVC project. Otherwise, the Web publishing wizard would be launched to perform Web deploys, and not the Windows Azure publishing wizard.

When you publish to Windows Azure for the very first time, you need to click on the “Sign in to download credentials” link on the publish page to download the .publishsettings file. This file contains the metadata and credentials needed for Visual Studio to work with your Windows Azure subscription. The .publishsettings download page will create a Windows Azure management certificate for Visual Studio and embed it in the .publishsettings file along with your subscription details. All of these details will be installed and stored on your local development machine when imported. *Important note:* This file contains very sensitive information such as subscription IDs and your management

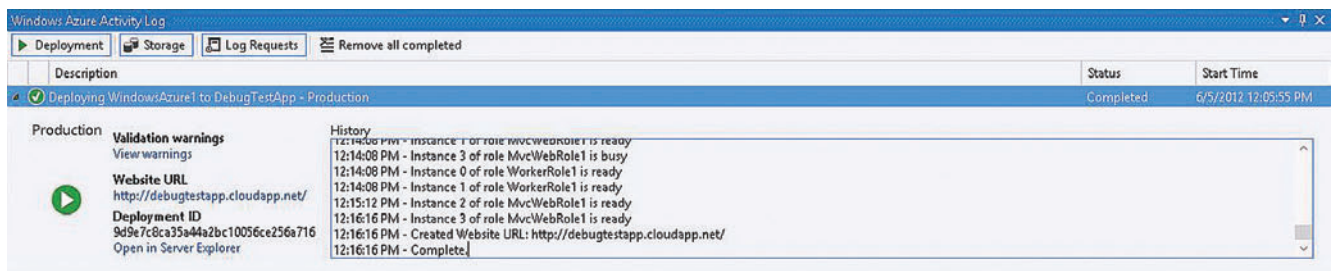


Figure 10 The Activity Log Window

SQL Server[®] LIVE!

TRAINING FOR DBAs AND IT PROS



Orlando, FL December 10-14

Royal Pacific Resort at Universal Orlando | sqllive360.com

SQL Server Live! provides comprehensive education on SQL Server database management, data warehouse/BI model design, Big Data, analytics, performance tuning, troubleshooting and coding against SQL Server.



Who Should Attend?

- Database Professionals (DBAs)
- Analytics Specialists
- Developers
- IT Professionals
- Consultants

Session Topics Include:

- BI and Big Data
- Hadoop
- T-SQL, SSIS Packages, Disaster Recovery
- Monitoring, Maintaining, Tuning
- SQL Azure & Windows Azure
- SQL Server 2012

Save Up To \$400!

**Register Before
October 10**

Use Promo Code DEVSQS

sqllive360.com



Buy 1 Event, Get 3 Free! Customize an agenda to suit your needs – access to all 4 Live! events is included in your registration. Attend just 1 conference or all 4 for the same price.

SQL Server Live! is co-located with:

Cloud & Virtualization LIVE!
THE FUTURE OF COMPUTING

SharePoint LIVE!
TRAINING FOR COLLABORATION

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

GOLD SPONSOR



SUPPORTED BY



MEDIA SPONSOR



PRODUCED BY



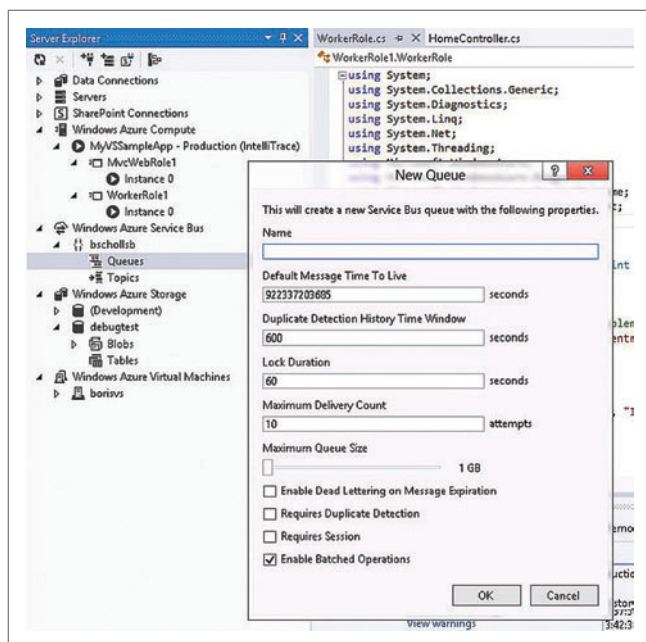


Figure 11 Creating a New Service Bus Queue

certificate, so it's best to store the file in a secure place or delete it immediately after importing.

Once the file is downloaded to the development machine, you can import it by using the Import button on the first page of the wizard. All of your subscriptions will now show up in the dropdown box, as shown in Figure 6. The nice thing about the dropdown box is that you can also manage your authentication settings such as creating new certificates, renaming credentials and so on.

Clicking the Next button takes you to the Common Settings tab (see Figure 7). This tab allows you to select an existing cloud service or even create a new one. Also, you can choose which environment you want to deploy to—production or staging—and which build service configuration you want to use. Remember we talked about multiple service configurations earlier in this article? This is the place where you can choose which ones to use for your deployment.

Remote Desktop and Web Deploy can be enabled on that tab as well. Enabling Remote Desktop is really helpful if you want to connect to specific role instances on Windows Azure later for diagnostic purposes. Once you've decided to enable Remote Desktop and the deployment has been successfully published, you can connect directly to the role instance using the "Connect using Remote Desktop" context menu on an instance in Server Explorer, as shown in Figure 8. The context menu of Server Explorer automatically displays menu items based on the functionality enabled during publish. In our example, we enabled Remote Desktop.

There are some useful settings in the Advanced Settings tab of the publishing wizard. Here you can specify which storage account to use for publishing or create a new

one. We recommend always creating or using storage accounts in the same location (that is, the datacenter) as the cloud service using them in order to avoid performance impacts caused by latency. In addition, you can control the update behavior of your deployment as well as the troubleshooting tools you want to use for your deployment.

Finally, after applying the settings you want to use for publishing, you'll be taken to the summary page of the wizard.

The summary page lists all the settings you made and allows you to save a target profile for publishing. Unlike the cloud configuration, the profile stores the Visual Studio publish configuration. The target profile is basically an MSBuild definition file with the file extension .azurePubxml. All the settings applied in the publishing wizard are saved to that file. This comes in handy when you have different publishing settings or multiple target environments. Just think about testing and production environments. In the test environment, you'll probably want to have IntelliTrace turned on, though in the production environment, you won't. Now, instead of going through the wizard again before you can publish to production, you just need to select the publishing profile for the production environment. This is quite effective with regard to standardizing publishing.

One last thing worth mentioning for publishing: Visual Studio will perform a one-time update of the connection strings for diagnostics and caching in the service configuration file by default with the value for the publish storage account if the value is still UseDevelopmentStorage=true, as shown in Figure 9. This behavior can be disabled in the Role Designer by unchecking the "Update development storage connection strings ..." checkbox.

Clicking the Publish button will kick off the publishing process after the build was successful. As publishing can take some time, it's useful that Visual Studio provides a detailed status of which publishing step is currently being executed in the Windows Azure Activity Log window. While there are several operations being handled by the background publish process, you still get a pretty good idea which step is currently being performed by looking at Activity Log status for publish. Figure 10 shows the Activity Log window after successfully publishing.

You can now browse to your cloud service in a browser by clicking the "Web site URL" in the window.

Working with Server Explorer to Access Windows Azure

We already looked at how to connect to one of your instances using Server Explorer. While connecting using Remote Desktop to Windows Azure Compute instances and Windows Azure Virtual Machines is certainly a big asset, it's worth pointing out some of the other useful features in Server Explorer. In the June 2012 SDK, you can connect directly to the Windows Azure Service Bus by providing the namespace and the key for that namespace. Once connected, you have visibility into Service Bus queues and topics. You can even create new Service

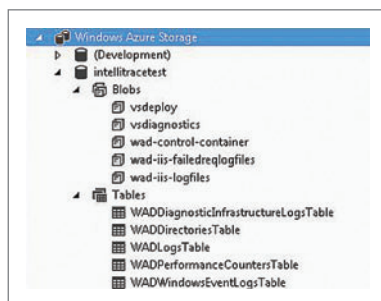


Figure 12 Storage Account Containing Diagnostics Data

Cloud & Virtualization LIVE!

THE FUTURE OF COMPUTING



Orlando, FL December 10-14

Royal Pacific Resort at Universal Orlando | virtlive360.com

Cloud & Virtualization Live! is a new event that provides in-depth training with real-world applicability on today's cloud and virtualization technologies.



Who Should Attend?

- Developers
- IT Professionals
- IT Managers
- Systems Administrators
- Consultants

Session Topics Include:

- Becoming a Virtualization Expert
- Constructing & Managing an Application Delivery Infrastructure
- Integrating Mobile Devices and Consumerization with Cloud Computing
- Managing the Modern Cloud Enabled Datacenter
- Tactics in Cloud Application Development

Save Up To \$400!

**Register Before
October 10**

Use Promo Code DEVCVS
virtlive360.com



Buy 1 Event, Get 3 Free! Customize an agenda to suit your needs – access to all 4 Live! events is included in your registration. Attend just 1 conference or all 4 for the same price.

Cloud & Virtualization Live! is co-located with:



Bus queues and topics, as shown in **Figure 11**.

Also, you can add storage accounts to the Windows Azure storage node and see what's stored in the BLOBs and tables. Currently, you can only read the data in Windows Azure storage using Server Explorer. Last, Server Explorer now also lists the Windows Azure Virtual Machines and all the endpoints exposed by the VMs.

Troubleshooting Options

Earlier you saw a code sample that initializes the diagnostics monitor to collect Windows event log data sources. Explaining how Windows Azure diagnostics works goes beyond the scope of this article; however, let's still have a quick look at it.

In addition to Windows event log data sources, you have the option to add code that will start collecting Windows Azure trace logs, infrastructure logs, crash dumps and so on. More information on enabling and configuring Windows Azure diagnostics can be found at bit.ly/MMkwik.

The major difference in collecting diagnostics data in Windows Azure versus an on-premises application is where the diagnostics data is stored. Windows Azure stores the diagnostics data in Windows Azure storage (we discussed the storage account for diagnostics earlier in the "Publishing to Windows Azure" section). Some diagnostics data, for example IIS logs and IIS failed requests, is stored in BLOB storage; other data, such as trace logs, performance counters and Windows event logs, is stored in table storage. **Figure 12** shows a storage account named `intellitracetest` with BLOB and table storage containing Windows Azure diagnostics data. The BLOBs and tables are easily identifiable, as they start with "wad-" or "WAD."

In order to move the data to the storage, it's important to always use the `ScheduledTransferPeriod` property. The following code shows an example for Windows event logs:

```
diagConfig.WindowsEventLog.ScheduledTransferPeriod = TimeSpan.FromMinutes(1.0);
```

Enabling diagnostics definitely helps with troubleshooting and monitoring. But, as developers, we love the debugger. Currently, Visual Studio doesn't provide out-of-the-box support for debugging, but you *can* use familiar tools such as IntelliTrace and Profiling. Before we wrap up, let's have a look at IntelliTrace. IntelliTrace is available only in the Visual Studio Ultimate editions, however, it's such an invaluable feature for cloud developers that it's definitely worth mentioning. Let's assume you need to check whether the Worker Role in your project started as expected. Because there's some tracing code in the `WorkerRole:RoleEntryPoint` class, you can use IntelliTrace to debug the `RoleEnvironment.OnStart` event.

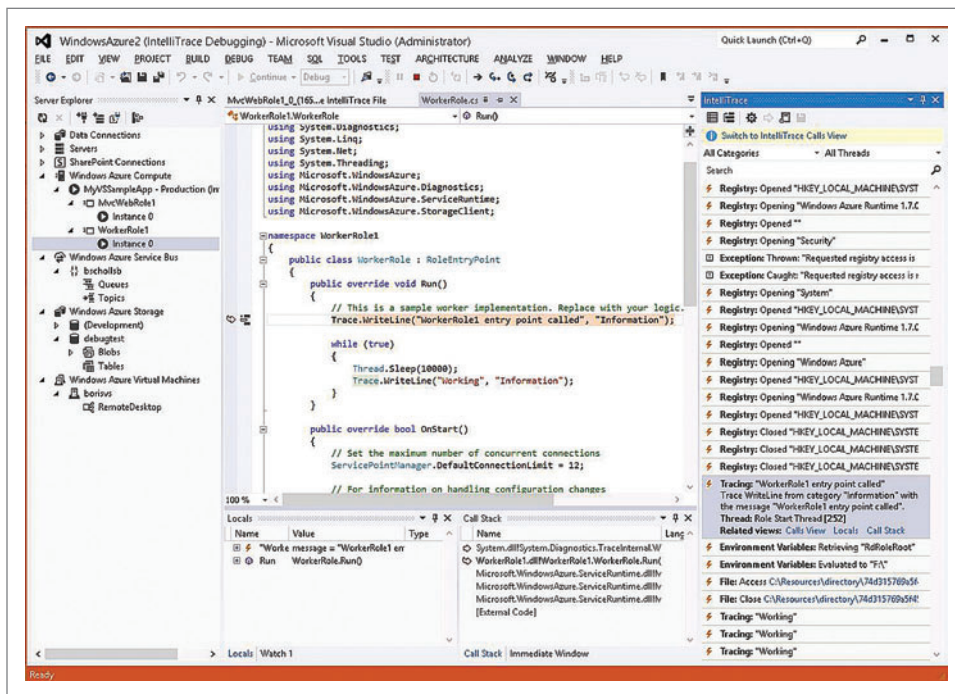


Figure 13 Debugging Using IntelliTrace

Remember that we previously discussed enabling IntelliTrace for your deployment? After the successful publish, when you right-click on a compute instance in Server Explorer and have IntelliTrace enabled for the deployment, you get another context menu item saying "View IntelliTrace logs" (it works the same way if you had enabled Profiling). Remember that enabling IntelliTrace and Profiling are mutually exclusive (a current limitation that we hope to remove in the future).

Once you click "View IntelliTrace logs," Visual Studio downloads the logs and displays them, as shown in **Figure 13**.

You're now able to see all of the events, and you can start debugging from the event in which you're interested. The IntelliTrace window gives good information on the sequence of the events, particular events and exception information. Visual Studio takes you straight to the relevant code in the code editor when you click on an event.

Even though we just scratched the surface of all the Windows Azure tools features, you should have an idea of how easy it is to develop and debug cloud service applications using Visual Studio. For the latest and greatest news and information about .NET development on Windows Azure, please see bit.ly/v5MF7m. ■

BORIS SCHOLL is a senior program manager with the cloud tools team for Visual Studio, focused on building end-to-end developer experiences for Windows Azure. Before joining the team, he spent time working on the Visual Studio SharePoint tools team and as an architect in the Microsoft field designing SharePoint and cloud solutions.

PAUL YUKNEWICZ is a principal program manager lead for cloud tools, Windows Forms and Visual Basic 6 for Visual Studio.

THANKS to the following technical experts for reviewing this article: Gordon Hodgenson, Jim Nakashima and Mohit Srivastava



YOUR .NET Resources



Visual Studio[®]
MAGAZINE

Visual Studio[®] **LIVE!**
EXPERT SOLUTIONS FOR .NET DEVELOPERS

ONLINE | NEWSLETTERS | PRINT | CONFERENCES

What's New in Microsoft Test Manager 2012

Sudheer Adimulam, Micheal Learned and Tim Star

In this article we'll introduce some of the new features in Microsoft Test Manager 2012 (MTM) that are used and “dogfooded” by the Visual Studio ALM Rangers.

To recap, the ALM Rangers are a group of experts who promote collaboration among the Visual Studio product group, Microsoft Services and the Microsoft Most Valuable Professional (MVP) community by addressing missing functionality, removing adoption blockers and publishing best practices and guidance based on real-world experiences.

Following are discussions of various features of MTM 2012.

Exploratory Testing This is sometimes referred to as “ad hoc testing,” defined as performing software testing without a defined script. The idea is to lean on the creativity of the tester to help surface bugs versus having a scripted step-by-step test case for every test run and scenario. In the first version of MTM, released in 2010, exploratory testing was enabled via filing an “exploratory bug” via

Microsoft Test Runner. The tool allowed users to perform a set of actions in an unscripted workflow, and once a bug was found, the tester could choose to trim down the recorded actions to an appropriate amount of steps to be included in the bug that was being filed.

The idea was that a tester might spend a considerable amount of time exploring the application before finding a bug, and then the ability to trim down the steps would give the tester the freedom to create the bug with more or fewer repro steps based on knowing the context in a given scenario. This functionality gave testers the ability to freely perform a test run in an unscripted style of workflow and still leverage the capabilities for creating a bug with the exact repro steps. Testers could also create test cases from these steps so the bug fix could be validated later by rerunning the scripted test case.

This article discusses a prerelease version of Microsoft Test Manager 2012. All related information is subject to change.

This article discusses:

- Exploratory testing
- Performance enhancements
- Video and audio recording
- Working with Visual Studio Team Foundation Server
- Testing Windows Store apps

Technologies discussed:

Microsoft Test Manager 2012

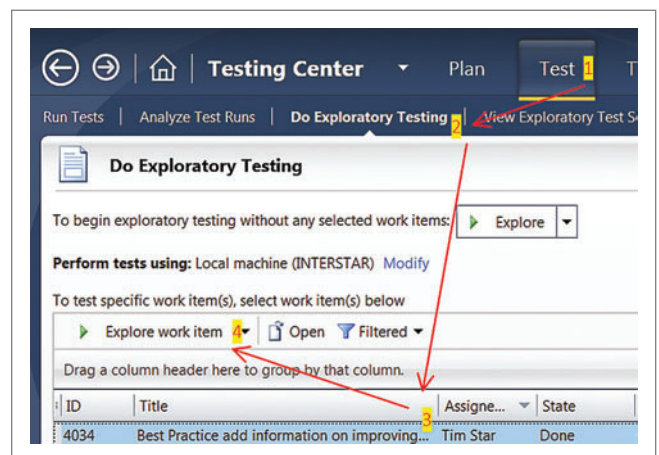


Figure 1 Steps for Exploratory Testing Using MTM

The exploratory testing experience has been greatly improved with the 2012 release. In the previous release, filing an exploratory bug first required having a test case to run from Microsoft Test Runner. Users could create a “dummy” test case, and, as an example, name it “Exploring” or just leverage some existing test case. Both of these options seemed somewhat clumsy and made discovering exploratory testing features somewhat difficult. In the 2012 release of MTM, a test case is no longer required for doing exploratory testing, and there are a few different ways to start exploratory testing sessions. To get started, simply right-click a test suite in the test plan and choose “Explore.” Users can also associate the exploratory testing effort with a requirement, which enables linking the bugs and test cases that are created to the requirement work items. To do this, launch the exploratory session from MTM 2012 from a backlog item, as shown in **Figure 1**.

While running the exploratory testing session, testers can create additional data for the bugs in the form of screenshots, comments and file attachments. The exploratory testing window shown in **Figure 2** provides a nice experience for testers. The large icons make it easy to create bugs and test cases, and you can type and format notes in a free-form field. The notes you type—and any data captured by testers, such as screenshots—get seamlessly added to the bug or test case as you create them. Users have the ability to add and remove steps to deviate from what was captured during the recorded actions.

Creating new bugs and test cases from an exploratory testing session is one example of a common workflow, but users can also open and update existing manual test cases and bugs.

Scenarios for Exploratory Testing in MTM 2012 The user experience is fast and fluid with exploratory testing in MTM 2012. Testers have the ability to pause and resume testing, which makes the overall user experience extremely flexible. One scenario used by many organizations to support software apps is to field customer phone calls, during which support agents often walk through the application with an end user to attempt to reproduce a bug. Often the support professional might take notes and screenshots and then later send the bug to the developers. The exploratory testing features of MTM 2012 theoretically enable support professionals to walk through the application via an exploratory testing session while on the phone with an end user and then feed

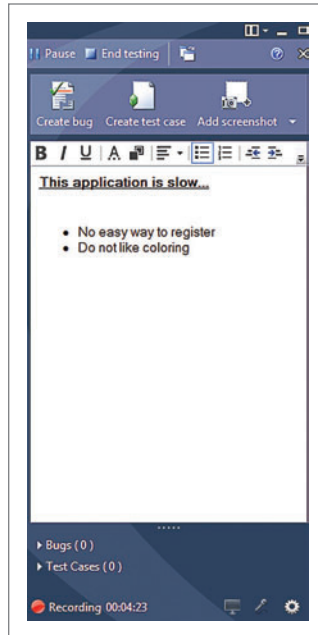


Figure 2 Exploratory Testing Session in MTM

the development teams the rich, actionable bugs efficiently. Once off the call, phone support professionals can then end the testing session and begin a new session with the next call.

Often, before releasing an application or new features to an application, product owners want to test the applications thoroughly to ensure bugs aren't released. By leveraging exploratory testing, organizations can remove the overhead of creating scripted test cases for every scenario. A group of testers could spend time exploring the application, and file bugs and test cases as they're found. This free-form testing can help reduce the overhead of more defined testing efforts.

For another example, users might want to test an application without the overhead of defined test cases simply because of temporary resource issues.

Improved Performance This was a major goal for the new release, and the product team has done a lot of work in this area. Connecting to a test plan, displaying tests within a suite, launching Microsoft Test Runner, saving work items and creating lab environments have all improved. In addition to

these improvements, Visual Studio Team Foundation Server (TFS) proxy support has been enabled for attachments so teams using MTM 2012 and TFS Proxy will now see performance benefits similar to those in source control operations. Specifically, attachments will be cached on the proxy server, saving each consumer of the attachment from having to wait for the attachment to be downloaded from TFS. A couple of other small timesavers include the addition of a “most recently used” list from which you can select a user without going through the complete list. Also, assigning configurations is made easier by providing a single list of configurations to choose from rather than a separate column for each.

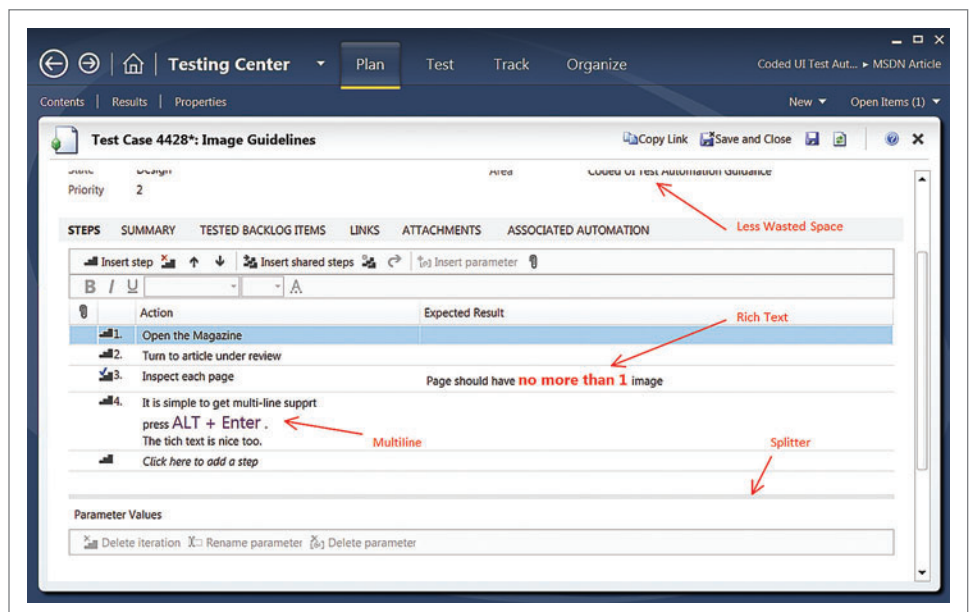


Figure 3 Test Case Editor Improvements

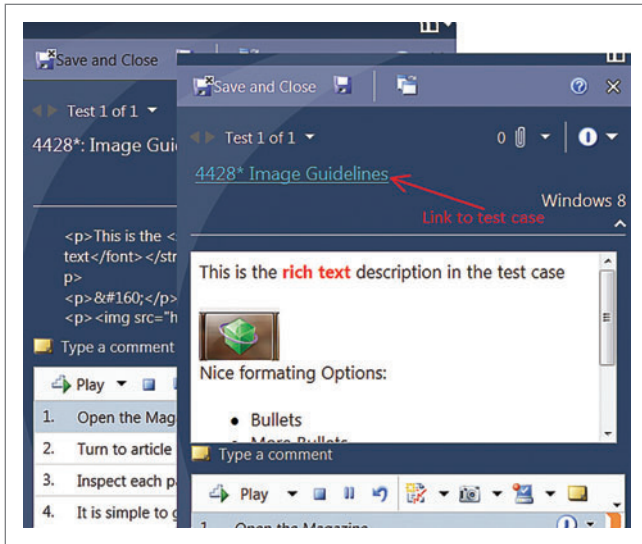


Figure 4 Microsoft Test Runner Enhancements

Test Case Editor Improvements Though it isn't obvious, the test steps grid (shown in Figure 3) has been completely rewritten. Features previously available via hotfix and feature packs are now available in the product by default. The test steps grid supports rich text and multiline test steps. Additionally, copy and paste from Microsoft Excel or Microsoft Word, including multiline steps and rich text, is supported. Screen real estate also is managed better by eliminating the frames around the test case fields at the top of the screen and by providing a splitter between the test steps grid and the parameters region at the bottom.

The video recorder no longer requires a separate install, and users may now optionally enable audio recording as well.

Cloning Test Suites into Other Plans for New Iterations A common question from MTM users has been, "How do I copy a test plan without losing traceability?" The 2010 version of MTM allowed test plans to be copied, which meant that a new test plan was created but new test cases were not created. Instead, the existing test cases were "referenced" by the new test plan. This meant that changing a test case in one plan also changed the test case in the other plan. This wasn't a desirable behavior for teams that required absolute traceability. Those teams had to use third-party utilities or resort to some low-level TFS API programming to achieve the desired results.

Cloning a test plan is now a feature of TCM.exe, the Test Case Management command-line tool. Cloning a test plan will clone the test cases, shared steps, test suites, assigned testers, configurations, action recordings, links, attachments and test configurations. Test settings, test results and test runs are not cloned. Also, requirement-based suites are not cloned. Cloning the original requirements and

associating them to new test cases or associating the new test cases to old requirements is a manual operation.

Performing a clone operation is done from TCM.exe within the Visual Studio command prompt. You must specify the collection, source and destination suites, and a value for the new destination test plan. You can optionally use the `overridefieldname` and `overridefieldvalue` parameters to specify a new area path or iteration path, or use custom test case fields that have been added to the test case work item template.

The `Tcm.exe suites` command format is as follows:

```
Tcm.exe suites /clone /collection:CollectionURL/teamproject:project /suiteid: id /destinationsuiteid: id /overridefieldname: fieldname /overridefieldvalue: fieldvalue
```

The following command line will copy a suite with an ID of 100 into a suite with an ID of 115:

```
tcm.exe suites /clone /collection:http://myTFS:8080/tfs/sampleTPC/teamproject:sampleTeamProject /suiteid:100 /destinationsuiteid:115 /overridefieldname:"Iteration Path" /overridefieldvalue:"areapath\sprint 2"
```

The team project collection is named "sampleTPC," and the team project is named "sampleTeamProject." The new iteration path will be "areapath\sprint 2."

Note: You can find the test suite ID by highlighting the test suite in the plan contents and then viewing the ID next to the suite name on the right-hand side in the header above the list of test cases.

Link to a Read-Only Version of a Test Case This is now provided in Microsoft Test Runner. Also, the test description field in Microsoft Test Runner supports rich text, as shown in Figure 4.

Video Recording Enhancements The video recorder no longer requires a separate install, and users may now optionally enable audio recording as well. Audio recording may be enabled or disabled in the Diagnostic Data Adapter for the video recorder, as shown in Figure 5.

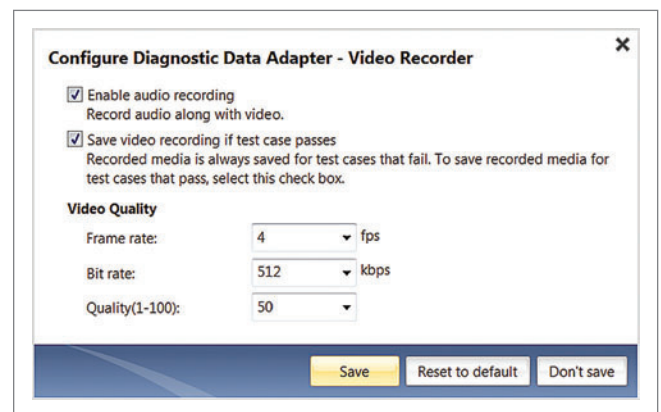


Figure 5 Enabling Audio Recording

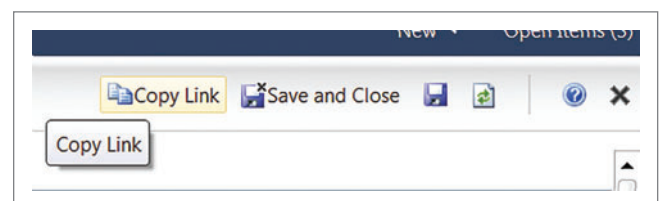


Figure 6 The Copy Link Button

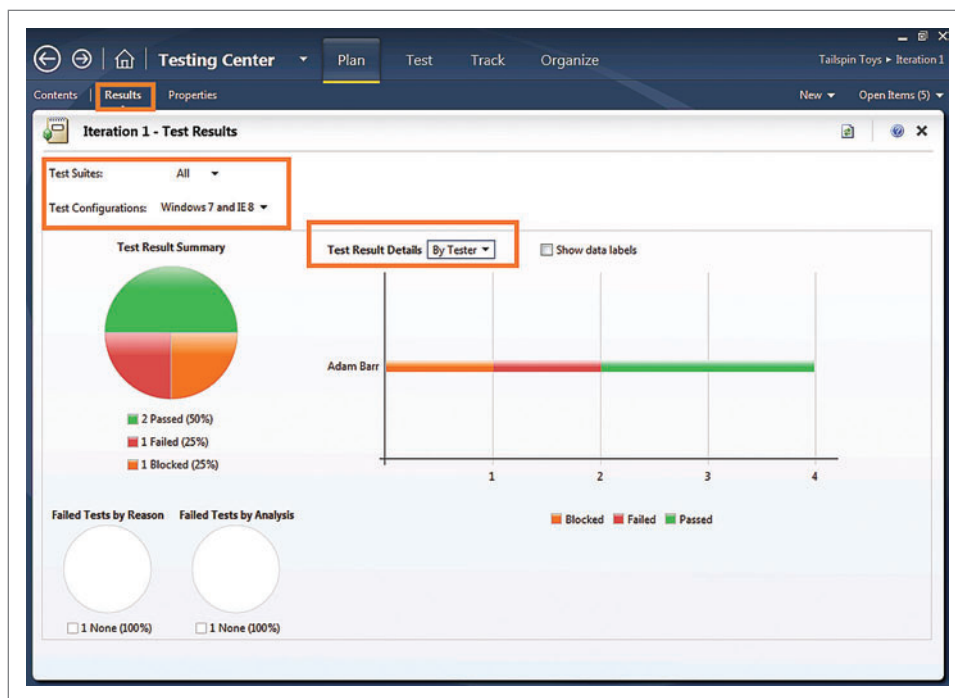


Figure 7 Test Plan Results

Navigation in MTM This has been improved in a couple of ways. You'll notice there's a Copy Link button sprinkled throughout the product, as shown in **Figure 6**.

Clicking this link will copy a URL to the clipboard so you can e-mail someone a hyperlink to the item you're viewing.

Clicking a hyperlink containing this address will launch MTM 2012 and bring the user directly to the test result identified with an ID and run ID, which are included in the hyperlink.

Test plan selection also has been improved. Selecting a plan from the launch screen could be painful in the previous version when a large number of plans were in the list. Rather than scrolling through the long list, simply type the first few letters in the plan to quickly jump to the appropriate location in the plan list.

There has always been a hyperlink in the upper-right corner of MTM that allows users to jump to the plan list. That feature still exists, but now there's a hyperlink to the team project as well. Jumping to a Team Projects selection screen is now also a single click away.

Connecting to TFS 2010 A majority of features of MTM—such as test planning and execution, data collection and use of lab environments—work fine between

mismatched versions of MTM and TFS. To use MTM 2012, you need to install TFS 2010 SP1 and the latest software updates. However, new features such as exploratory testing won't work until you upgrade TFS 2010 to TFS 2012.

Reports MTM provides various kinds of reports to track and measure the effectiveness of testing. The reports help you figure out which test cases have been passed, failed or blocked. MTM 2012 allows you to view the results from the Plan tab. There's an option to view results, which gives a good view of the test plan result status. You can view the results based on the test configuration or based on the test suite for which you want to see the results. In addition to this, users can also see the results based on the tester. To view the results in the Plan tab, click the Results link as shown in

Figure 7. This will open up the results for the most recent test run.

Test Data Reduction to Reduce Load on TFS Storage In MTM 2010, by default, when the results of automated test runs are published to TFS 2010, deployment items and binaries of all the test runs are uploaded. These can be used later to rerun tests and analyze failures. This approach has a large overhead in terms of TFS database storage and performance issues on the client side

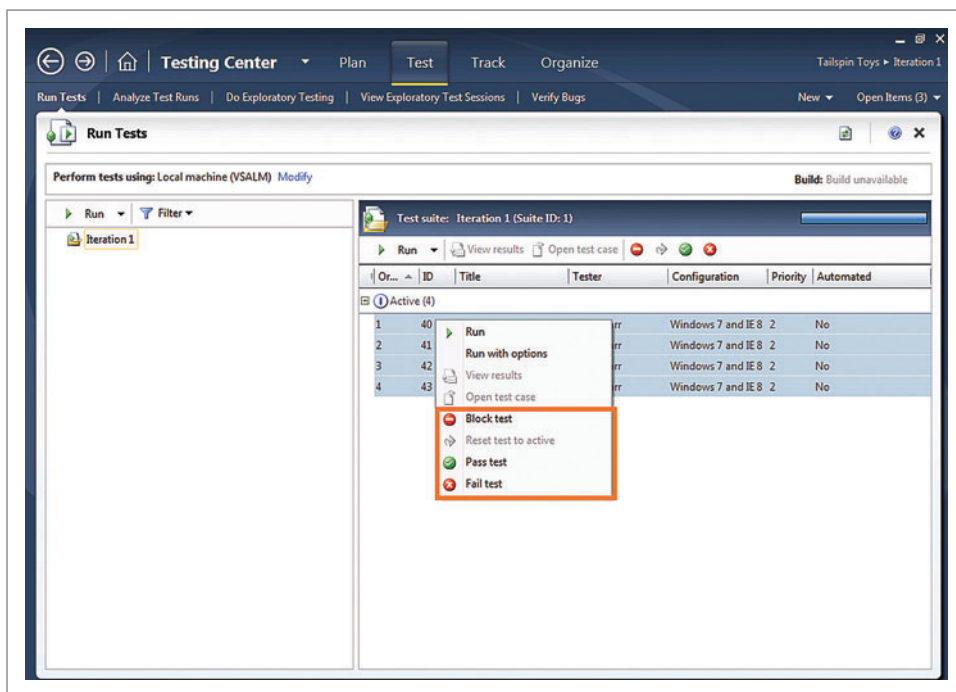


Figure 8 Mark Test Case Results in MTM

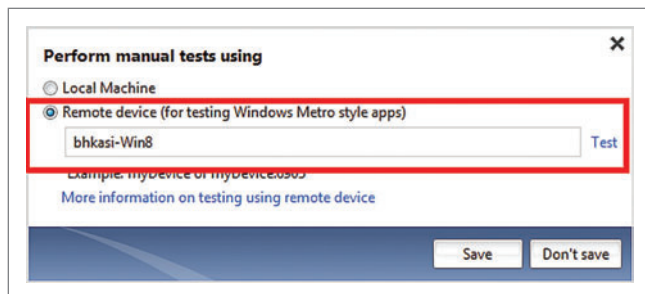


Figure 9 Manual Testing of Windows Store Applications

while opening the test results. In MTM 2012, by default, only the test result files and other data collector attachments are uploaded to the TFS database. Only when code coverage or test impact analysis is enabled are the binaries uploaded to TFS 2012. These binaries are required for code analysis.

Marking Test Case Results in MTM Without Launching Microsoft Test Runner In MTM 2010, there is no option for marking multiple test cases as pass or fail. The tester is only able to set the status of a single test case, and this has to be done from the Microsoft Test Runner window, which is a tedious job. With MTM 2012, testers can mark a test case pass, fail or block directly from the Run Tests screen of the Test tab. The tester also has the option to mark a single test case or multiple test cases as Pass test, Fail test or Block test, or Reset the test to active (see Figure 8).

Manual Testing of Windows Store Applications MTM 2012 helps improve the efficiency of manual testing of Windows Store apps. Using MTM 2012, you can test Windows Store applications that are running on a remote Windows 8 device such as a tablet or Windows 8 PC. You can execute your test steps on the remote Windows 8 device and at the same time mark the steps as pass or fail in MTM 2012 on your local machine. MTM 2012 will help you generate rich action logs—with a video and both text and image descriptions of your actions—that are step-by-step representations of the actions you performed on the remote device.

Manual testing of Windows Store applications consists of three steps. The first step is installing Remote Debugger, which consists of the Microsoft Test Tools Adapter service. The second step is connecting to the remote device using MTM 2012. The third step is executing the test cases from MTM 2012.

Before testing Windows Store applications, ensure the Microsoft Test Tools Adapter service is enabled. Once the service

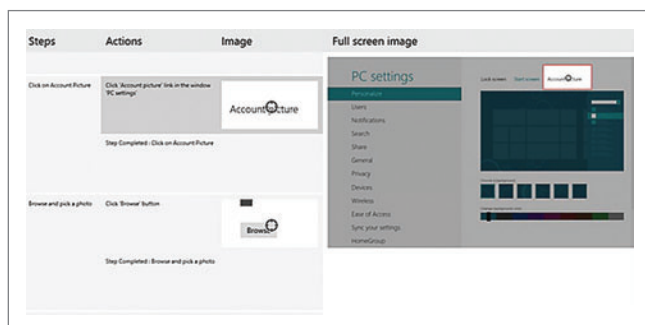


Figure 10 Enhanced Action Logs for Windows Store Apps
(Source: Visual Studio ALM + Team Foundation Server Blog at bit.ly/NVOEru.)

is enabled, in MTM 2012, connect to the test plan where you have your test suite. In the Testing Center, click the Modify link next to “Perform manual tests using” to specify the remote device on which to run manual tests (see Figure 9). Select the “Remote device...” option and enter the name or IP address of the device you want to test. Click Test to test the connection and then save your changes.

Once the connectivity is established, you can run the manual test case. Microsoft Test Runner opens up a “Perform manual tests using” dialog box with options to either Start Test or Install Application. Install Application will do a remote installation of the Windows Store app on the Windows 8 device, which is a three-step process of copying files, installing certificates and installing the app. Clicking the Start option will display the test steps in the MTM window, where you can mark them as pass or fail. While executing the steps in the remote machine, you can take screenshots of the bugs and create bugs.

MTM 2012 helps
improve the efficiency
of manual testing of
Windows Store applications.

Enhanced Action Logs for Windows Store Apps With MTM 2012, you generate rich action logs with both text and image descriptions of the actions performed on the Windows Store applications or Internet Explorer 10. The action log files contain screenshots for each action step conducted during the test run, and the files are saved as .html files and can be viewed in the browser. Hovering on any thumbnail in the image action log will display a full-screen image of the action performed (see Figure 10). The enhanced action log makes reproducing bugs easier. The user can see the exact steps taken by the tester, and these logs are displayed when a bug is submitted through Microsoft Test Runner or the exploratory testing window.

Be sure to explore these and many more features in MTM 2012, especially if you’re responsible for raising the quality bar of solutions and testing them. ■

SUDHEER ADIMULAM is a test consultant with Microsoft Services – Global Delivery, and works as a Visual Studio ALM Ranger. He has a master’s degree in computer applications and is an ISTQB, CSQA, MCSD and MCTS.

MICHAEL LEARNED is a senior premier field engineer developer with Microsoft and works as a Visual Studio ALM Ranger. He focuses on helping Microsoft customers with .NET Framework development and application lifecycle management. He can be reached at his blog at tfsmentor.com or on Twitter at twitter.com/mlhoop.

TIM STAR is a principal consultant with Intertech Inc., focusing on training, consulting and Visual Studio ALM. He has a bachelor’s degree in electrical engineering and is an MCPD, MCTS, MCT, Visual Studio ALM External Ranger and three-time MVP award recipient.

THANKS to the following technical experts for reviewing this article:
Mathew Aniyar, Nivedita Bawa, Willy-Peter Schaub and Charles Sterling

MSDN Magazine Online

The screenshot shows the MSDN Magazine Online website. At the top, there's a navigation bar with links: Home, Topics, Issues and Downloads, Script Junkie, Subscribe, Submit an Article, and RSS. A search bar is also present. The main content area features a large header with the MSDN logo and the text "THE MICROSOFT JOURNAL FOR DEVELOPERS" and "AUGUST 2012 VOL 27 NO 8". Below this, there's a section for "scriptjunkie" which is now on MSDN magazine. The main article preview is for "Windows Azure: Windows Azure Comes to the Rescue" by Mark Kromer. To the right, there's a sidebar with a "Columns" section featuring "Data Points" by Kenny Kerr. On the far right, there's an advertisement for "ScaleOut StateServer Version 5!" with a "Free Trial" button. The bottom of the page has a "Features" section highlighting C++ 11 standard improvements.

MSDN Magazine

Search MSDN Magazine with Bing

United States - English Sign in

Home Topics Issues and Downloads Script Junkie Subscribe Submit an Article RSS msdn

THE MICROSOFT JOURNAL FOR DEVELOPERS

AUGUST 2012 VOL 27 NO 8

msdn magazine

scriptjunkie is now on msdn magazine Read More

Windows Azure: **Windows Azure Comes to the Rescue**

What do you do when marketing comes to you needing a fully functional conference registration system for a show just a month away? You leverage Windows Azure, Silverlight and Windows Phone to build an app with all the social media trimmings, as Microsoft's Mark Kromer details here.

Mark Kromer

Windows Azure

Windows with C++: Lightweight Cooperative Multitasking

Kenny Kerr describes a technique to write asynchronous code in C or C++ without the use of a runtime.

Kenny Kerr

Announcing ScaleOut StateServer® Version 5!

Free Trial

SCALEOUT SOFTWARE

MSDN Magazine Blog

Standing Desk Experiment

Like a lot of developers, my job has me spending a LOT of time sitting in front of a computer monitor. So much so that I'm

Features

C++

C++: Functional-Style Programming in C++

The new C++ 11 standard greatly improves the language's suitability for

Columns

Data Points

Pitfalls and Pointers for a Base Logging Class in EF Models

An unfortunate use of a base

It's like *MSDN Magazine*—only better. In addition to all the great articles from the print edition, you get:

- Code Downloads
- The *MSDN Magazine* Blog
- Digital Magazine Downloads
- Searchable Content

All of this and more at msdn.microsoft.com/magazine



Testing for Continuous Development

Larry Brader and Alan Cameron Wills

Web applications are typically updated or extended every few weeks (or even days) in reaction to shifting business needs and customer feedback. To support continuous development, every aspect of the development cycle must be more efficient and lightweight than more traditional development processes. For example, it's in the nature of software that even the smallest change requires everything to be retested. But repeating a full suite of manual tests every few days is impossible. This means that tests eventually must be automated, even if they begin as manual explorations. In this article we show how Microsoft Visual Studio 2012 supports testing in a continuous development environment.

This article discusses a prerelease version of Visual Studio 2012. All related information is subject to change.

This article discusses:

- Testing in the DevOps cycle
- Testing in Visual Studio 2012
- Automated testing and manual testing
- Automating manual tests
- Lab management
- Reports in Visual Studio Team Foundation Server
- Visual Studio 2012 and continuous development

Technologies discussed:

Visual Studio 2012, Visual Studio Team Foundation Server, Microsoft Test Manager

The DevOps Cycle

Few software projects come to an end when the software is deployed and running. Feedback is obtained from the stakeholders, improvements or extensions are planned, and eventually a new version is released, whereupon the cycle starts again. This process, known as the DevOps cycle, is illustrated in **Figure 1**.

In a traditional software project, each cycle can take years. The first release typically is rich in features, delivered on a DVD and installed on users' local machines. By contrast, in a modern Web application, the first release might be minimal, but extensions and improvements are released every few weeks (or even days).

For example, the managers of a social-networking site might try a new feature for a week, during which they monitor how customers use it. At the end of the trial period, they adjust the details of how the feature works.

In this much more rapid cycle, it's important for development teams to think about the DevOps cycle as a whole process. In particular, they need to make each activity in the cycle more efficient, removing any roadblocks that hinder progress around the loop.

The improvements in the testing process we discuss here are all aimed at reducing the time it takes to cycle through the DevOps loop. In particular, these new tools and techniques aim to reduce the bottleneck that testing has traditionally caused in this loop.

Testing in the DevOps Cycle

The essential role of testing is to demonstrate that user stories and other requirements have been implemented. The most effective way to do this is to run the application manually and exercise each

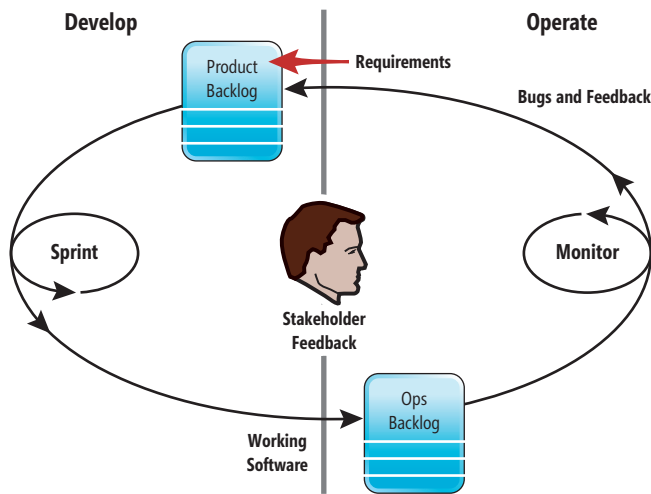


Figure 1 The Continuous Development Cycle

story just as the end users will. An experienced tester applies a variety of strategies to expose bugs and explores all the variants and edge cases of the application's behavior.

When application code changes, it's prudent to retest everything that might depend on it. Dependencies in software typically form a complex weave, and bugs notoriously turn up in features that seem unrelated to the focus of the update. For this reason, traditional development teams don't like changing any component that has been written and tested. As noted, the slightest change demands a full retest, so if you test everything manually, this requires a great deal of effort and resources.

In contrast, the short cycles inherent in continuous development require that each part of the software be frequently revisited as its functionality is improved and extended. It would be impossible to manually retest every feature every few days. Short cycles require automated tests, which replace the manual tests with program code. You can run coded tests quickly and as often as you like.

The essential role of testing is to demonstrate that user stories and other requirements have been implemented.

Should continuous development teams code all their tests and abandon manual testing altogether? That approach has sometimes been suggested, but in practice, there's an efficient compromise, which works as follows.

Test new and substantially changed stories manually. When each test passes consistently, create automated versions of the tests for that story. In consequence, although the total number of tests gradually increases as the product expands, the load of manual testing remains constant because manual testing is something you do only for relatively new features.

In fact, the bulk of coded tests in a typical continuous development project are *unit tests*, which are written along with the application code and test individual components inside the software, rather than testing the behavior of the whole application. Unit testing is a powerful tool for maintaining stability as the code base is updated.

Figure 2 shows a gradual transition from manual to automated tests over time, together with the expansion of unit tests along with the application code. The diagram is an ideal picture; in practice, most teams automate only some proportion of their manual tests. Visual Studio 2012 (and other versions) also provides for partial automation, which can be used to speed up tests without writing code.

Testing in Visual Studio 2012

Let's see how testing is supported by Visual Studio 2012 and its associated products, Visual Studio Team Foundation Server (TFS) and Microsoft Test Manager (MTM).

If your specialty is testing the whole application, you'll be more interested in the support provided by MTM. If you're a developer, you might take more interest in the support for automated testing in Visual Studio 2012. However, continuous development demands a closer relationship between these two roles, and some teams dispense with the distinction altogether. Therefore, the Visual Studio 2012 tools are designed to integrate the different styles of testing, and they support a broad spectrum of testing practices, from the more traditional approaches through continuous development.

Automated Testing with Visual Studio 2012

Automated testing includes all types of tests that are defined by writing or generating program code. You create automated tests in Visual Studio 2012, where you initially run them for debugging.

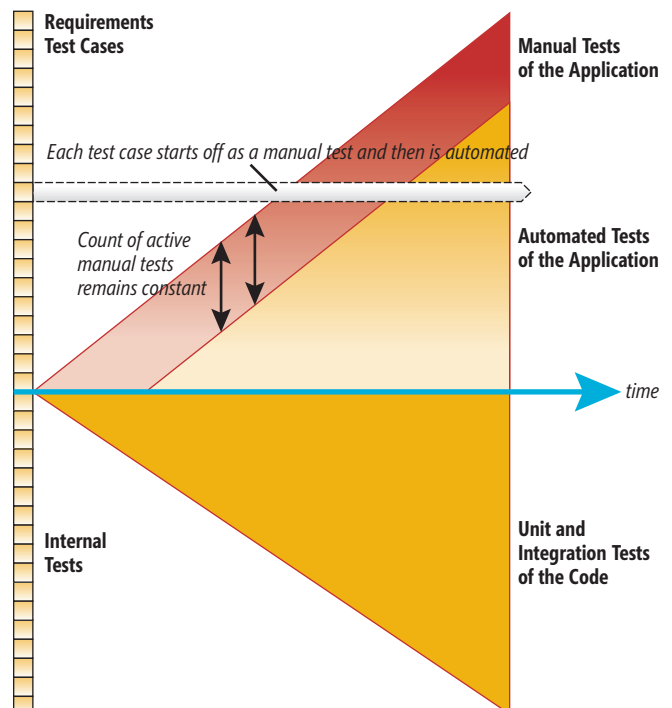


Figure 2 Ideal Transition from Manual to Automated Tests over Time

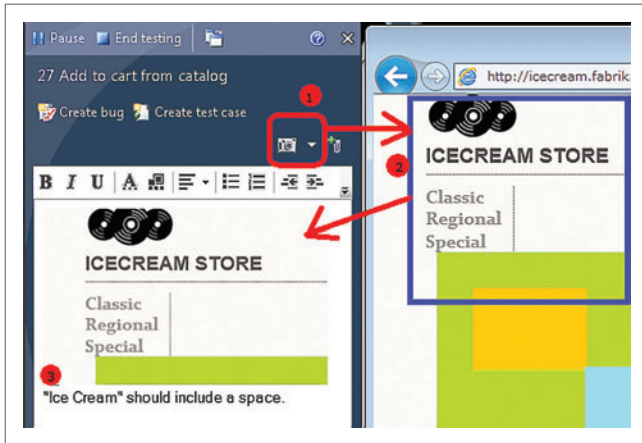


Figure 3 Recording a Screenshot and Making Notes in the Exploratory Testing Window

When the test—and the application code that it tests—are correct, you check in the test, along with the application code. From the source code repository, it's picked up by the build service and run regularly according to your team's build definitions.

Unit and Integration Tests Unit testing is one of the most effective ways of maintaining a bug-free codebase through successive changes in an application.

A unit test is a method that tests a method, class or larger component of your application in isolation from other parts, external systems and resources. In practice, developers often write *integration tests*—that is, tests written in a similar way to unit tests but which might depend on external databases, Web sites or other resources. Either way, these tests use the same tools and infrastructure.

In Visual Studio 2012, you can write tests that use any of several test frameworks, such as NUnit, xUnit and the default VSTest. When you have coded tests in any of these frameworks, you simply open the Test Explorer window and choose Run All. The test results are summarized in the window.

Background testing is an option that efficiently runs your tests in the background every time you build your solution. The tests affected by your changes are performed first. This means that as you work, you can constantly see which tests are passing or failing.

Isolate Units by Using Fakes True unit testing means disconnecting the unit under test from the code on which it's dependent. This has a number of advantages. If your unit is being developed or updated at the same time as other units on which it's dependent, you can test it without waiting for the others to be complete. If you restructure the application to use this unit in a different way, or in a different application, the tests go with it and don't need to change.

Visual Studio 2012 provides two mechanisms, collectively called *fakes*, for disconnecting a unit from its dependencies. Calls from your unit to methods outside its boundary can be handled by small pieces of code that you provide. For example, you can define a *shim* that intercepts calls to any external method such as `DateTime.Now`. Because it always receives the same response from the shim, your unit will demonstrate the same behavior every time it's invoked. You can also define *stubs*, which provide placeholder implementations of methods in assemblies that haven't been loaded.

The Visual Studio 2012 tools are designed to integrate the different styles of testing, and they support a broad spectrum of testing practices.

Performance and Load Tests Visual Studio 2012 Ultimate provides specific test facilities for performance and stress testing. An application can be instrumented and driven so as to measure its performance under specified loads. Web applications can be driven with multiple requests, simulating many users.

Coded UI tests let you run your application and generate code that drives its UI. Visual Studio 2012 includes specialized tools for creating and editing coded UI tests, and you can also edit and add to the code yourself. For example, you might create a simple procedure to buy something at a Web site and then edit the code to add a loop that buys many items.

Coded UI tests are particularly useful where there's validation or other logic in the UI—in a Web page, for example. You can use them either as unit tests for the UI or as integration tests for the whole application.

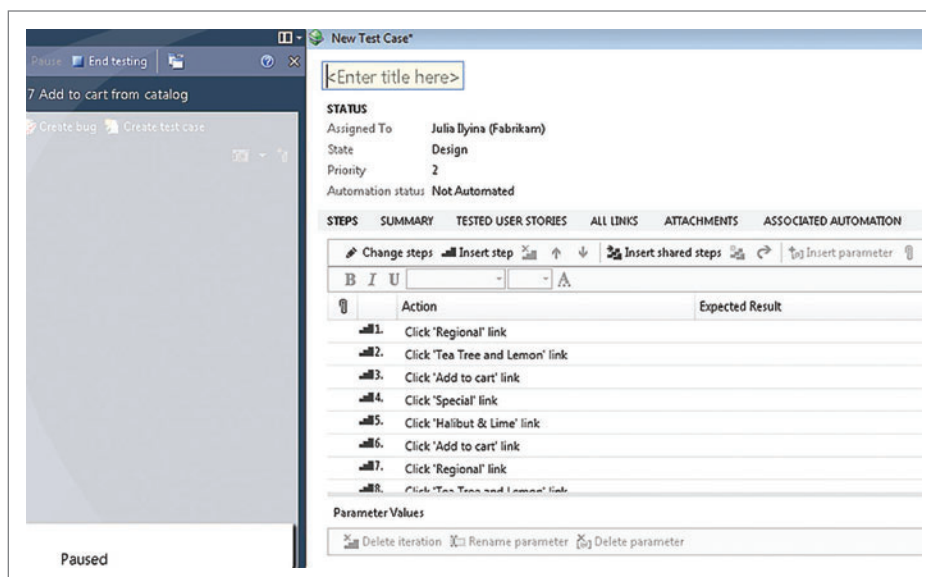


Figure 4 Defining Steps and Expected Results in a Test Case

Manual Testing with Microsoft Test Manager

Manual testing can be *planned* or *exploratory*. You perform manual tests with the help of MTM. Tests are normally performed on versions of your application that have been built from checked-in code.

Typically, manual tests are linked to user stories (or product backlog items or other requirements), and the results of the tests are displayed in reports on the project dashboard. This means that everyone can quickly see which stories have been successfully implemented.

Exploratory Testing Exploratory testing simply means running the application to try it out. However, why do you need MTM to help you run it?

MTM can record your actions, comments and screenshots while you work. If you decide to create a bug report, all this information is automatically added to it, making it unnecessary for you to add a precise description of how to reproduce the bug. **Figure 3** shows an example of the MTM exploratory testing window alongside a Web application that's being tested.

MTM can also instrument the application itself, both in the client and the server, and record event data that can be used to debug the application. This data is automatically attached to your bug report.

When the bug is fixed, you'll want to repeat the steps you took in the exploration to verify the fix. To help with this, you can generate a test case from the exploratory session, in which you include the relevant steps.

Planned Testing with Test Cases Test cases are manual tests that you define as a series of steps the tester should perform. **Figure 4** shows the steps defined in a test case.

Test cases provide a great way to clarify what the users need. At the start of the sprint, when you're discussing stories or requirements with the users and other stakeholders, you can use the steps as a precise example of what the users will be able to do by the end of the sprint. Each test case is just one instance of the requirement, and so each requirement is usually associated with more than one test case. For example, if the requirement is to be able to buy ice cream, one test case will detail the steps to buy a particular flavor. You would create another test case to describe buying a mixture of flavors. The guiding principle of the discussion with stakeholders should be: "When you can successfully perform these test cases, then we'll consider the story to be implemented."

In TFS, both the stories or requirements and the test cases are represented by work items. You can link them together so the progress of a requirement can be tracked by the results of the tests.

When you run a test case, the steps are displayed at the side of the screen. You check off each step while you run the application. At the end, you check off whether the test has passed or failed.

Just as with exploratory testing, your actions, comments, screenshots and application data are recorded so you can create a detailed bug report very quickly.

A great advantage of using steps is that they help anyone repeat the test reliably, even if they aren't familiar with the application.

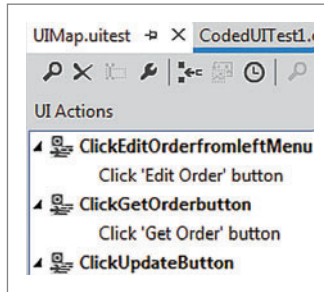


Figure 5 Editing UI Actions in Visual Studio 2012

When a test is repeated, you can be confident that whether it passes or fails, it isn't simply because the test was run differently from the last time.

You can also generate a planned test case from an exploratory session. Doing this helps ensure that you always run the test using the same actions.

Automating Manual Tests

Automating a substantial portion of your manual tests is essential to minimize the time taken by testing in the DevOps cycle. Visual Studio 2012

supports this automation in several ways.

Record/Playback You can rerun a test case semiautomatically. On the second and subsequent times you run a test, MTM replays the keystrokes and gestures that you used on the first run. All you have to do is verify that the results you see conform to the expectations detailed in the steps.

Automating a substantial portion of your manual tests is essential to minimize the time taken by testing in the DevOps cycle.

Playback makes manual testing quicker and more reliable. It also makes it possible to distribute the testing load among colleagues who might not be completely familiar with the application.

Even if you don't fully automate a test, rapid and reliable playback helps reduce the DevOps cycle time. This feature does not require Visual Studio 2012 to be installed and does not involve writing code.

Generating Coded UI Tests You can generate a fully automated coded UI test from a recorded manual test case run. The generated code performs the same actions as the manual test. By using a special editor in Visual Studio 2012, you can also extend the test to verify the results and generalize it to repeat the test for different input data. **Figure 5** shows the special editor.

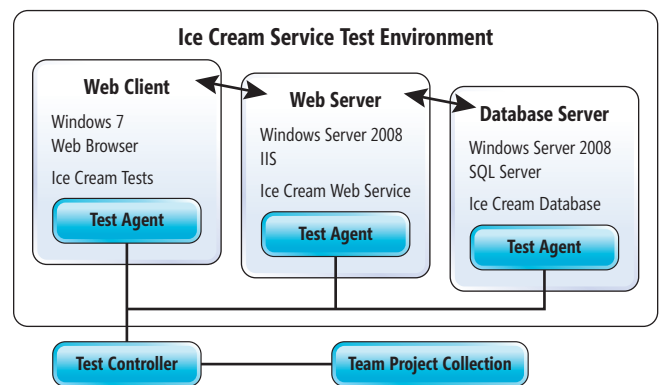


Figure 6 A Sample Lab Environment for Testing a Sales Web Site

Figure 7 User Story Test Status Report

Title	Work Progress		Test Status			
	% Hours Completed	Hours Remaining	Test Points	Test Results		Bugs
Customers can buy ice cream.	60%	16	3	67%	33%	3
Customers can select flavor from catalog.	60%	10	3	67%	33%	
Vendor can vary flavor catalog.	25%	15	2	100%		
Vendor can set different prices for flavors.	39%	23	2	100%		4
Customers can set favorite flavors.	100%	0	1	100%		4
Users can choose U.K. spelling of favourite flavour.	100%	0	6	50%	33%	
Customers can select different types of cones.	75%	5	22	63%	26%	2

Linking Test Cases to Test Methods You can link a test case to any test method, even if it hasn't been generated from your test run. The result of running the test will be reported as if you had run the manual steps. Typically you would link the test case to an integration test that performs the same actions as the manual test but drives the business logic directly, rather than using the UI.

This approach has the benefit that changes in the UI layout don't invalidate the test. It's also useful when the development team has already created a suitable integration test.

Lab Management

When you test an application, the first thing you need is a machine to run it on. In fact, most applications today need several machines. For a realistic test environment, you might, for example, need to install a Web server, a database server and a client browser all on separate computers. **Figure 6** illustrates such an environment.

In addition to the basic installation, you'll also want to install agents that can collect the event data that was mentioned earlier in the "Exploratory Testing" section.

In MTM a feature named Lab Center makes all of this straightforward. Lab Center lets you define lab environments. A lab environment is a set of machines that will be used as a group for test purposes.

In addition to handling the assignment of the machines (so you don't accidentally use a machine that's running someone else's

tests), Lab Center also installs the necessary test agents. Lab Center provides a console where you can quickly log in to any of the machines in an environment.

Lab Center is also good at creating and managing virtual machines (VMs). You can create a virtual environment, install the relevant platform software and then store a library copy of it that you can use whenever you want to test your application. All you have to do is reinstantiate a clean copy of the environment and install the new versions of the application's components. You can also automate this deployment process.

Using Lab Center—and, in particular, taking advantage of its facility with VMs—can significantly decrease lab setup time compared to more traditional approaches to maintaining a lab. Lab Center is a substantial contributor to reducing the time spent in each DevOps cycle.

Automated Testing on TFS

Automated tests are initially performed in Visual Studio 2012 on a developer's computer. After the code has been checked in to the source repository, there are a number of ways in which tests on the integrated code can be run by the build service.

Periodic Builds The build service compiles the code and runs the tests. You can create build definitions to specify which tests should be run, and you can specify when they should run. For example, you might run a core set of tests on a continuous basis and run a more extensive set every night.

Build results can be viewed in Visual Studio 2012 and are also available from your project's TFS Web service. E-mails can notify you of failures.

Lab Deployment As we previously described, you can assign a group of lab machines to a test by using Lab Center. By defining a lab build, you can automate this process. When your build is triggered—for example, when code is checked in, or at a particular time of day—the build starts by compiling all the application and test code. If this is successful, a lab environment is assigned, and if it's a virtual environment, it can be set back to a fresh state. Your application components are then deployed to the correct machines, and the tests are installed on the designated client machine from where they drive the application.

The tests can be automated tests of any kind, but typically you use this type of build to perform large integration tests or tests of the whole application.

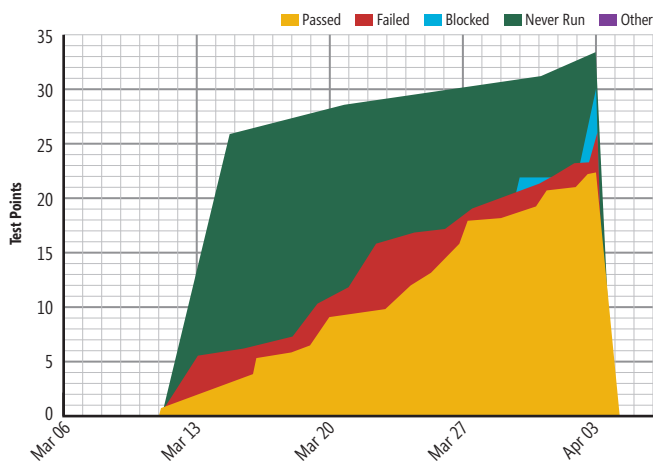


Figure 8 Test Plan Progress Report for a Sprint

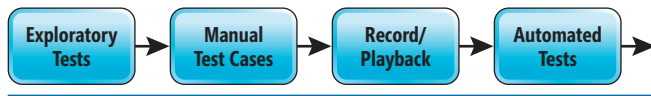


Figure 9 Test Progression

If your tests are linked to test cases, the results will be recorded against the related user stories or requirements and displayed in the project progress reports.

Reports

TFS provides a number of charts and tables that show the progress of a project. You can view them either individually or in project dashboards. Among the reports are several related to testing.

For example, the User Story Test Status report, illustrated in **Figure 7**, shows the list of stories you're working on in the current sprint. In addition to the development work performed for each story, the chart shows the success or failure of its associated tests. If failures were discovered while running the tests, the resulting bug reports are also linked to the requirements.

The results on the chart come from both the most recently run manual tests and the automated test runs.

The DevOps cycle views development as just one half of a process that also incorporates feedback from operations.

The Test Plan Progress report, illustrated in **Figure 8**, shows how many test cases were created for the current sprint and how many have been run.

Some teams like to create test cases at the start of each sprint, as a target for the team. All the tests should be green at the end of the sprint.

Visual Studio 2012 and Continuous Development

You now have some familiarity with the range of the Visual Studio 2012 testing features, from unit tests to whole-application manual tests.

The DevOps cycle views development as just one half of a process that also incorporates feedback from operations. Depending on the context, the DevOps cycle will be short or long. If you're developing a nuclear power station, hopefully you'll go around the loop very slowly. If you're running a Web application, you might go around it every few days. Slow and fast cycles are equally valid and appropriate for different types of systems. Both incorporate the need for testing, to different degrees. If the fast cycle of continuous development is appropriate for your project, it's important to reduce the time it takes to perform every action in the loop. You'll probably also make less distinction between the roles of developer and tester than in projects that work at a more measured pace.

The tools available in Visual Studio 2012 can substantially reduce the amount of time it takes to test your application. Here are some points to remember:

- Clean lab environments can be set up quickly and automatically, especially by using VMs. Take advantage of the Lab Center.
- Recording actions during exploratory and planned testing lets you create bug reports quickly and reliably, reducing the chance that a bug will disappear when someone tries to reproduce it.
- You can implement a gradual transition from manual tests to automated tests. The automated tests handle the bulk of regression testing while manual tests focus on new and updated stories.
 - From exploratory tests, you can generate repeatable manual test cases.
 - You can record test runs and play them back rapidly and reliably.
 - You can generate coded UI tests from test runs. Alternatively, you can link separately coded integration tests to test cases.

Figure 9 shows the progression from exploratory tests to automated tests.

- Test cases can help you describe your user stories precisely. You can work with your stakeholders to write the steps to reduce any misunderstandings about what the stories do.
- Tests are linked to requirements (or user stories or product backlog items) so you can see comprehensive reports on how far the users' needs have been implemented—not simply in terms of the work done on them, but in terms of whether they're successful. The tests provide the goal for the development team in each sprint.
- Test cases and requirements can be linked to storyboards in PowerPoint, requirements documents or Unified Modeling Language models. When you make changes in the storyboard, you can trace the necessary changes to the tests.
- A test plan links together test cases, code, requirements, test environments and test settings, and the team project. If the team spends a few months working on something else before returning to enhance a product, it's easy to reconstruct everything required for testing.

We've given you an overview of how Visual Studio 2012 fits in with the DevOps cycle. You should now understand why you need a streamlined approach to testing and what tools Visual Studio 2012 offers to help you fulfill this goal. If you want more information, read "Testing for Continuous Delivery with Visual Studio 2012 RC" in the MSDN Library at bit.ly/KHd0q4. It's an in-depth guide that covers every aspect of the testing infrastructure provided by Visual Studio 2012. Related articles include "Verifying Code by Using Unit Tests" at bit.ly/dz5U3m and "Testing the Application" at bit.ly/NbJ01v. ■

LARRY BRADER has been a senior tester on the Microsoft patterns & practices team for the past several years. Before that he worked as a developer and tester on military and medical technologies.

ALAN CAMERON WILLS is a programming writer in the Microsoft Developer Division. In previous lives he has been a developer, a software architect and a consultant in development methods.

THANKS to the following technical experts for reviewing this article: Howie Hilliker, Katrina Lyon-Smith, Peter Provost and Rohit Sharma

Shape up Your Data with Visual Studio LightSwitch 2012

Jan Van der Haegen

Visual Studio LightSwitch was initially released in mid-2011 as the easiest way to build business applications for the desktop or the cloud. Although it was soon adopted by many IT-minded people, some professional developers quickly concluded that LightSwitch might not be ready for the enterprise world. After all, a rapid application development technology might be a great way to start a new and small application now, but, unfortunately, history has proven that these quickly created applications don't scale well once they grow large, they require a lot of effort to interoperate with existing legacy systems, they aren't adaptive to fast-paced technological changes, and they can't handle enterprise requirements such as multitenant installations or many concurrent users. On top of that, in a world of open standards such as RESTful services and HTML clients being adopted by nearly every large IT organization, who would want to be caught in any closed format?

This article discusses a prerelease version of Visual Studio LightSwitch 2012. All related information is subject to change.

This article discusses:

- Designing data
- Shaping data
- Exposing data
- Building client applications

Technologies discussed:

Visual Studio LightSwitch 2012

With the release of LightSwitch 2012 coming near, I thought it was a perfect time to convince some friends (professional developers and software architects) to put their prejudices aside and (re)evaluate how LightSwitch handles these modern requirements.

The outcome? It turns out LightSwitch 2012 doesn't just handle all of these requirements, it absolutely nails them.

Understanding Visual Studio LightSwitch

LightSwitch is a designer-based addition to Visual Studio 2012 to assist in working with data-centric services and applications. When working with a LightSwitch project, the Visual Studio IDE changes to a development environment with only three main editors (in so-called "Logical mode"): the Entity Designer, the Query Designer and the Screen Designer. These editors focus on getting results quickly by being extremely intuitive, fast and easy to use. This adds some obvious benefits to the life of a LightSwitch developer:

- First, it hides the plumbing (the repetitive code that's typically associated with the development of these information systems). Easy-to-use editors mean fast development, fast development means productive developers and productive developers mean more value to the business. Or, as Scott Hanselman would say, "The way that you scale something really large, is that you do as little as possible, as much as you can. In fact, the less you do, the more of it you can do" (see bit.ly/lnQC2b).
- Second, and perhaps most important, nontechnical people, ranging from functional analysts to small business owners to Microsoft Access or Microsoft Excel "developers"—often

referred to as citizen developers—who know the business inside out, can step in and help develop the application or even develop it entirely. The editors hide technological choices from those who prefer to avoid them and silently guide the application designer to apply best practices, such as encapsulating the domain logic in reusable domain models, keeping the UI responsive by executing business logic on a thread other than the UI thread, or applying the Model-View-ViewModel (MVVM) development pattern in the clients.

- Finally, these editors actually don't edit classes directly. Instead, they operate on XML files that contain meta-data (LightSwitch Markup Language), which is then used by custom MSBuild tasks to generate code during compilation. This effectively frees the investment made in business logic from any technological choices. For example, a LightSwitch project that was made in version 1.0 would use WCF RIA Services for all communication between client and server, whereas that same project now compiles to use an Open Data Protocol (OData) service instead (more on OData later). This is about as adaptive to the ever-changing IT landscape as an application can get.

Designing Data

Designing data with LightSwitch can be considered equivalent to creating domain models, in the professional developer's vocabulary. The IDE first asks you about "starting with data," specifically where you want to store this data. Two possible answers are available. You can choose to "Create new table," in which case LightSwitch uses the Entity Framework to create the required tables in SQL Compact (when debugging the application), SQL Server or Windows Azure SQL Database. Or you can choose to design your entities over an existing data source, such as SharePoint, an OData service, an existing database or a WCF RIA Services assembly. The latter two can really help you keep working on an existing set of data from a legacy application, but expose it through modern and open standards in an entirely new service or application.

As an example, you can take an existing OData service (an extensive list of examples is available at odata.org/ecosystem) and import one or multiple entities into a new LightSwitch application.

Name	Type	Required
EmployeeID	Integer	<input checked="" type="checkbox"/>
LastName	String	<input checked="" type="checkbox"/>
FirstName	String	<input checked="" type="checkbox"/>
Title	String	<input type="checkbox"/>
TitleOfCourtesy	String	<input type="checkbox"/>
BirthDate	Date Time	<input type="checkbox"/>
HireDate	Date Time	<input type="checkbox"/>
Address	String	<input type="checkbox"/>
City	String	<input type="checkbox"/>
Region	String	<input type="checkbox"/>
PostalCode	String	<input type="checkbox"/>
Country	String	<input type="checkbox"/>
HomePhone	String	<input type="checkbox"/>
Extension	String	<input type="checkbox"/>
Photo	Binary	<input type="checkbox"/>
Notes	String	<input type="checkbox"/>
PhotoPath	String	<input type="checkbox"/>
Employees1	Employee Collection	<input type="checkbox"/>
Employee1	Employee	<input type="checkbox"/>
<Add Property>		<input type="checkbox"/>

Figure 1 The Entity Designer After Importing from an Existing OData Service

Figure 1 shows you how the Employee entity from the Northwind data source is first represented in the Entity Designer. With a few clicks and a minimal amount of coding effort where needed, you can redesign the entity in several ways (the end result can be found in Figure 2). These include:

Name	Type	Required
EmployeeID	Integer	<input checked="" type="checkbox"/>
LastName	String	<input checked="" type="checkbox"/>
FirstName	String	<input checked="" type="checkbox"/>
Title	String	<input type="checkbox"/>
TitleOfCourtesy	String	<input type="checkbox"/>
NameSummary	String	<input type="checkbox"/>
BirthDate	Date	<input type="checkbox"/>
HireDate	Date	<input type="checkbox"/>
Address	String	<input type="checkbox"/>
City	String	<input type="checkbox"/>
Region	String	<input type="checkbox"/>
PostalCode	String	<input type="checkbox"/>
Country	String	<input type="checkbox"/>
HomePhone	Phone Number	<input type="checkbox"/>
Extension	String	<input type="checkbox"/>
Photo	Image	<input type="checkbox"/>
PhotoPath	Web Address	<input type="checkbox"/>
Notes	String	<input type="checkbox"/>
TeamMembers	Employee Collection	<input type="checkbox"/>
Supervisor	Employee	<input type="checkbox"/>
<Add Property>		<input type="checkbox"/>

Figure 2 The Same Employee Entity After Redesign

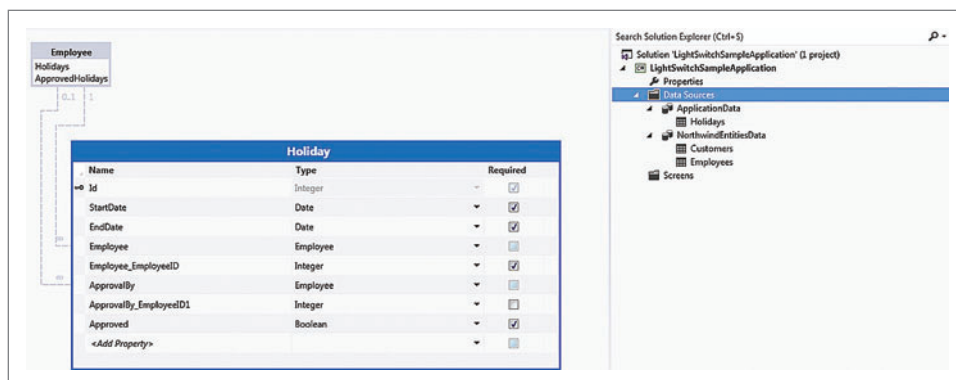


Figure 3 Virtual Relationships over Different Data Sources

- Renaming properties or reordering them (TeamMembers and Supervisor).
- Setting up choice lists for properties that should only accept a limited number of predefined values (City, Region, PostalCode, Country, Title and TitleOfCourtesy).
- Adding new or computed properties (NameSummary).
- Mapping more detailed business types over the data type of a property (BirthData, HireDate, HomePhone, Photo and PhotoPath).

LightSwitch understands a large set of commonly used business types such as E-mail Address and Phone Number, and more can be found in the many available extensions, or you can create your own business types via a LightSwitch extensibility project. With respect to the underlying data type, business types augment a property with specific characteristics such as specialized validation (for example, checking the E-mail or Web Address formats) and specific extended properties (Phone Number formats for Phone Number business types), and will often come with specialized controls used (by default) to represent or interact with the property in the client apps.

Entities are hardly ever totally unrelated. Using the Entity Designer, you can design relationships between different entities where needed. These relationships will be enforced on all layers—through code in the client and middle tier, and through foreign keys in the database when designing entities that will be stored in an intrinsic (a “new”) data source, such as SQL Compact, SQL Server or Windows Azure SQL Database.

However, a truly powerful feature of LightSwitch is that you can define new relationships on existing data sources as well. This is especially handy when trying to move forward with existing sets of data, such as XML files or old and malformed databases without indexes, keys or proper relationships. These so-called virtual relationships will be enforced in the client and on the middle tier without actually having to alter the legacy data source. Even more powerful is that these virtual

relationships can be defined between entities in different data sources.

For example, by right-clicking on Data Sources in Solution Explorer and selecting Add Table, you can create a new Holiday entity that will be stored in a new database, which has virtual relationships with the Employee entity from the Northwind OData service (see Figure 3).

Creating a holiday tracker application this way is a great example of how LightSwitch helps you break the boundaries of using data in just

one application to provide additional value in another.

Shaping Data

LightSwitch tries to keep the amount of code needed to get started to an absolute minimum, which speeds up initial development tremendously. Before a data service truly is ready to be exposed, it will often need some tweaks and modifications to shape the business inside the data. This is often easier to express in code than through the editor.

For every imaginable event, LightSwitch has an extension point that can be accessed from the Write Code button in the designer (see Figure 4). Depending on whether your code will be called on the server, in the client or on both tiers, this will generate a method stub where you can implement your custom code in either the client, common or server subproject of a LightSwitch project.

Continuing the example of the holiday tracker application, you could use these code extension points to initialize an entity. For example, the following code will automatically assign an employee’s supervisor as the person to approve the holiday, or approve the holiday if the person has no supervisor:

```
public partial class ApplicationDataService
{
    // Initializing a new Holiday - server only
    partial void Holidays_Inserting(Holiday entity)
    {
        if (entity.Employee.Supervisor != null)
            entity.ApprovalBy = entity.Employee.Supervisor;
        else
            entity.Approved = true;
    }
}
```

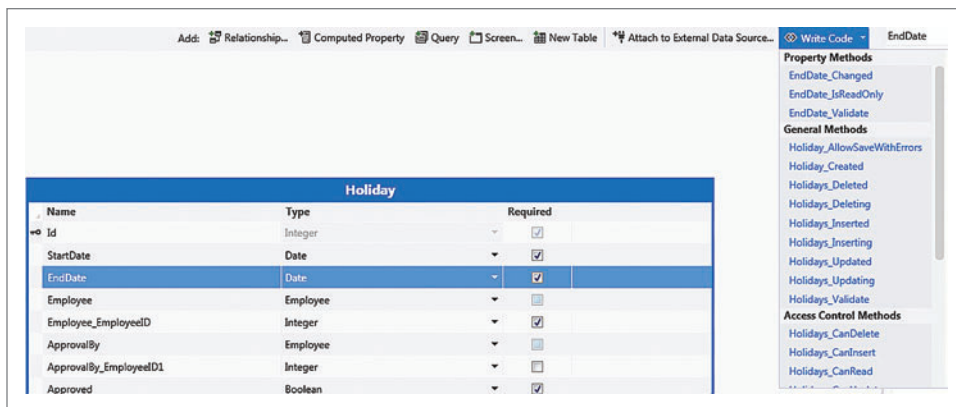


Figure 4 LightSwitch Supports Many Extension Points



Figure 5 Using the Query Designer for a Simple Query

Similarly, these code extension points can help you write custom validation code beyond what the business type of the property already validates:

```
public partial class Holiday
{
    // Determine if an "EndDate" is valid - executes client and server
    partial void EndDate_Validate(EntityValidationResultsBuilder results)
    {
        if (StartDate > EndDate)
            results.AddPropertyError("End date must follow the start date");
    }
}
```

Besides custom validation and business rules, LightSwitch also ships with a built-in, optional access control model based on roles and permissions. Checking access control can be done from these code extensibility points as well, both vertically and horizontally.

Vertical access control is where you'd impose limitations per screen, entity or property based on the permissions of the currently logged-in user. For example, the following code would limit the approval process to only those employees within a particular department of the organization, such as human resources:

```
public partial class Holiday
{
    // Determine if "Approved" can be modified by the user -
    // executes client and server
    partial void Approved_IsReadOnly(ref bool result)
    {
        result = !Application.User.HasPermission("ApproveHolidays");
    }
}
```

Horizontal access control is where you'd filter which records are visible to the end user. Note that this code runs on the server tier, meaning data that's not allowed to be accessed by the end user is never sent through the wire:

```
public partial class ApplicationDataService
{
    // Filtering out records - runs on the server only
    partial void Holidays_Filter(ref Expression<Func(Holiday, bool)> filter)
    {
        if (!Application.Current.User.HasPermission("ViewAllHolidays"))
            filter = holiday => holiday.Employee.NameSummary ==
                Application.Current.User.FullName ||
                Application.Current.User.FullName == holiday.ApprovalBy.NameSummary;
    }
}
```

Horizontal access control, also called row-level security, is an important addition to LightSwitch in Visual Studio 2012.

First, the filter is executed on the server. This restricts the amount of data that can be needlessly transferred per call because the user isn't

allowed to interact with these rows in the first place. In vertical access control, the end user will still see the data, even if he can't interact with it because the controls are read-only or exceptions are thrown in the case of violating actions. However, in the case of horizontal access control, the excess data is completely hidden from the end user.

This brings us to another powerful use for row-level security: It can grant ownership of parts of the entire data set only to particular users or organizations. This allows you to create an application that only needs to be installed once, and you can provide access to different groups, companies, individuals or organizations. Each of them acts as a tenant in that they all use the same client application and the same services, but only own a slice of the data. These multitenant installations tremendously lower hosting costs for all parties and offer other benefits such as the ability to update just once in one central location, such as the cloud. Those who had the pleasure of spending hour after hour dialing in to on-site servers to update installation after installation of an urgent hotfix will probably be the first to acknowledge the power of these multitenant capabilities in LightSwitch 2012.

Exposing Data

When thinking about the value of the data a company collects, many tend to think about it only in terms of the product that uses



Figure 6 The LightSwitch Publish Wizard

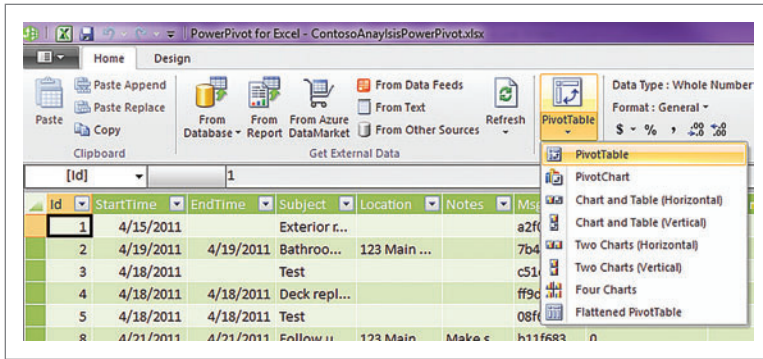


Figure 7 Consuming a LightSwitch OData Service in Excel

the data. However, collecting and analyzing broad categories of data is becoming more valuable than ever. When freed from the silo of a single application and treated correctly, analyzing and mining data can provide additional revenue beyond the traditional product-based business model. For this to work, it's important to embrace open standards, such as the OData protocol that's used on the service that's automatically built when building a LightSwitch project.

Here's a summation of OData from odata.org:

"OData, short for the Open Data Protocol, is a Web protocol for querying and updating data ... OData does this by applying and building upon (open) Web technologies such as HTTP, Atom Publishing Protocol and JSON ... OData is consistent with the way the Web works—it makes a deep commitment to URIs for resource identification and commits to an HTTP-based, uniform interface for interacting with those resources (just like the Web). This commitment to core Web principles allows OData to enable a new level of data integration and interoperability across a broad range of clients, servers, services and tools."

In other words, OData is implementing a service that can be interacted with through the use of simple HTTP verbs and resource identifiers (or URLs). An OData service can be consumed from almost any client technology, and in its simplest use, this means the service can even be browsed with an HTTP "Get" request from a Web browser, for example, by typing the following in as the Web address:

```
http://services.odata.org/Northwind/Northwind.svc/Customers?$filter=
replace(CompanyName, '%20', '') eq 'Alfreds Futterkiste'
```

Such a URL is always composed of the service root URL (the endpoint of the service), a resource path (the name of the entity) and, optionally, query options. Note that the latter makes the service itself quite agnostic of the use case; the OData protocol has an extensive query language that can be used to sort or filter data via the URL however the caller needs it. You can find the full set of operators at bit.ly/LSiPAj.

When you're designing the LightSwitch data service and you're aware of how the data will be consumed, you can also take advantage of the Query Designer to create queries before publishing, for your own convenience. **Figure 5** shows a simple query that returns

only the unapproved holidays that start during the current year. Note how the source of the query is the entire Holidays set, which will, of course, be prefiltered by the filter I set in the code earlier.

After spending some time in the Entity Designer and Query Designer, you can have an OData service ready to be deployed. This can be done directly from the Visual Studio IDE by right-clicking on the LightSwitch project in Solution Explorer and choosing the "Publish..." option from the context menu.

This brings up the LightSwitch Publish Application Wizard (see **Figure 6**), with which you can set up some final application-specific properties such as required connections strings, authentication types (none, username and password combination, or Windows authentication), an application administrator account and the destination of your installation (which can be an installation package or entail directly publishing to IIS or Windows Azure from the IDE). This last option especially has seen some tremendous improvements over the past couple of months, which makes deploying your data service to the cloud a fast and worry-free experience.

By adopting an open standard, the set of clients that can be used to interact with this data service is one that almost every end user will have available already. Spreadsheet applications such as Microsoft Excel allow the data service to be consumed and turned into PowerPivot tables and graphs without any coding required (see **Figure 7**). (Note: PowerPivot, built-in to Office 2013, is a separate, free add-on to Office 2010.) You can find a step-by-step walk-through about this at bit.ly/xETGOV.

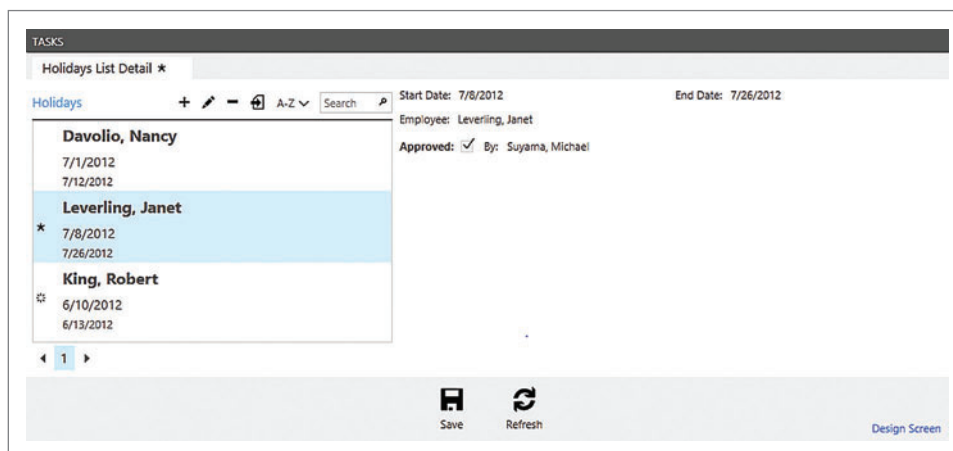


Figure 8 A Simple LightSwitch Client Application

Building Clients

Another key benefit of LightSwitch is that you can use the same IDE experience to build client applications. LightSwitch supports both thick clients (Silverlight 5, in-browser or out-of-browser) and mobile companion applications (HTML5) with the same design speed it provides on the server side, or even faster. Note that the ability to create HTML5 apps is only available in the preview release, but will be available as an add-on after the final Visual Studio 2012 release.

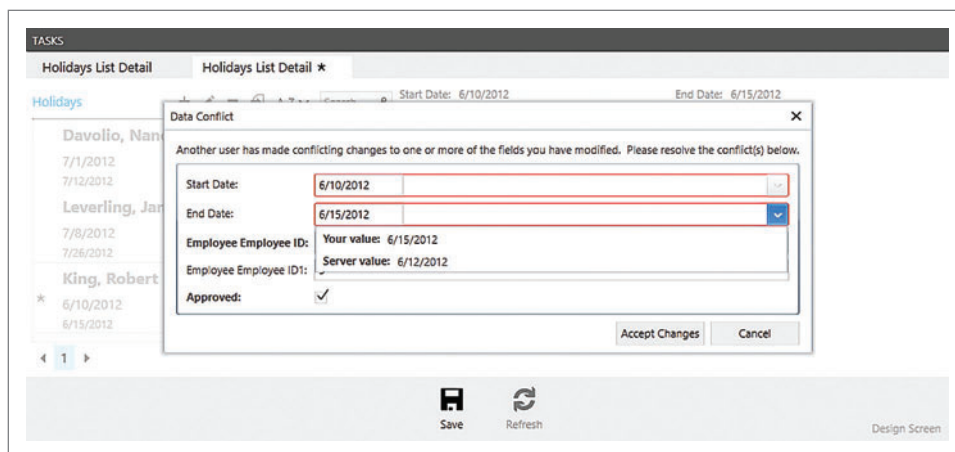


Figure 9 Handling Concurrent Modifications

To get started, click the Add Screen button in either the Entity Designer or Query Designer. This will bring up a wizard where you can select a screen template, which can either be one of the many built-in templates or one you've created yourself or downloaded from the available extensions.

Selecting your screen data (the Holidays entity set) and a template (list and details screen) will quickly generate the desired screen layout and open it in the screen designer. You can then start with some initial modifications to the layout of the controls or change which controls are used entirely. LightSwitch does a good job of inferring the desired control to use based on the business type of any property, such as using a date picker for a birth date property, but you're free to change this to another LightSwitch control, either one supplied out of the box or downloaded from available extensions. Or you can let your XAML (or JavaScript and CSS) skills run wild for unlimited customization. At this point, the application is ready to be built and run with a click of the F5 button.

You no longer need
to worry about raising
NotifyPropertyChanged events
yourself to get your bindings to
function correctly.

As you can see in **Figure 8**, the application is clean, intuitive and fully functional. However, don't let the ease with which it was created trick you into believing it's anything but complete. For example, under the covers, it uses an advanced version of MVVM, based on runtime interpretation of metadata, which I covered in an online April 2012 article, "The LightSwitch MVVM Model" (msdn.microsoft.com/magazine/hh965661). Because of this, the Screen Designer that was used inside Visual Studio to do the initial design can be brought up inside the application by clicking the

design screen link at the bottom right, so you can make further modifications to the view metadata while the application is running in debug mode. These modifications can then be persisted back to the project.

While further exploring the LightSwitch client architecture, you'll find there are many more technological wonders to discover. For example, controls or screens will automatically become read-only or hidden if you lack the permission to edit the underlying entity.

Equally convenient is that you no longer need to worry about raising `NotifyPropertyChanged` events yourself to get your bindings to function correctly. If the end user changes an employee entity's family name property, for example, a control bound to a computed `fullname` property (based on the family name property, of course) will automatically be updated to display the new value.

Speaking of changing a property: All of the models used in the client are actually "dual-dispatcher objects" to help keep the UI responsive while processing business logic. From the UI thread, you can do a simple assignment of a value to a property. This will internally send a notification that's picked up on a weighted priority basis in a second thread, called the logical thread. Once the business logic has finished processing, the same event-notification system is used to inform the UI thread of the eventual end result and update the UI accordingly.

One last interesting fact I wanted to share about the client comes from the very active Visual Studio LightSwitch Team Blog, where it's explained that each screen actually represents a unit of work (see bit.ly/9vENdF). Each screen has its own data workspace, which it shares with any child screen opened by that initial screen, such as pop-ups or an entity's detail screen. All changes are kept locally until the end user decides to save or discard all of them. This means that by opening a screen twice (allowing multiple instances is disabled by default but can be enabled in the Properties window from the screen designer), you can simulate how LightSwitch handles different users making concurrent modifications on some data and then saving them at the same time (see **Figure 9**).

If you're wondering how well LightSwitch handles this, as I said in the introduction, it doesn't just handle all of these requirements, it absolutely nails them. ■

JAN VAN DER HAEGEN is a green geek who turns coffee into software. He's a loving husband, a proud scrum master of an international team at Centric Belgium, and so addicted to learning about any .NET technology—Visual Studio LightSwitch in particular—that he maintains a blog on his coding experiments. You can find his latest adventures at switchtory.com/janvan.

THANKS to the following technical experts for reviewing this article:
Joe Binder and Beth Massi



Coding Logistic Regression with Newton-Raphson

Logistic regression (LR) is a machine-learning technique that can be used to make predictions on data where the dependent variable to be predicted takes a value of 0 or 1. Examples include predicting whether or not a patient will die due to heart disease within a certain number of years (0 = not die, 1 = die), based on the patient's age, sex and cholesterol level, and predicting whether or not a baseball team will win a game (0 = lose, 1 = win) based on factors such as team batting average and starting pitcher earned run average. Logistic regression assumes that problem data fits an equation that has the form $p = 1.0 / (1.0 + e^{-z})$ where $z = b_0 + (b_1)(x_1) + (b_2)(x_2) + \dots + (b_n)(x_n)$. The x variables are the predictors and the b values are constants that must be determined. For example, suppose you want to predict death from heart disease. Let the predictor variables be x_1 = patient age, x_2 = patient sex (0 = male, 1 = female) and x_3 = patient cholesterol level. And suppose you have somehow determined that $b_0 = -95.0$, $b_1 = 0.4$, $b_2 = -0.9$ and $b_3 = 11.2$. If there's a 50 year old male patient whose cholesterol level is 6.8, then $z = -95.0 + (0.4)(50) + (-0.9)(0) + (11.2)(6.8) = 1.16$, and so $p = 1.0 / (1.0 + \exp(-1.16)) = 0.7613$. The p value can loosely be interpreted as a probability, so in this case you'd conclude that the patient has a 0.7613 probability of dying within the specified number of years.

When using logistic regression, the primary problem is how to determine the b (often called beta) values for the LR equation.

The function $1.0 / (1.0 + e^{-z})$ is called the sigmoid function. The domain of possible values for z is all real numbers. The result of the function is a value between 0.0 and 1.0 as shown in **Figure 1**. You can't assume that the data you're working with can be modeled by the sigmoid function, but many real-life data sets can in fact be accurately modeled by the function.

When using logistic regression, the primary problem is how to determine the b (often called beta) values for the LR equation. In most situations, you have some historical data with known results

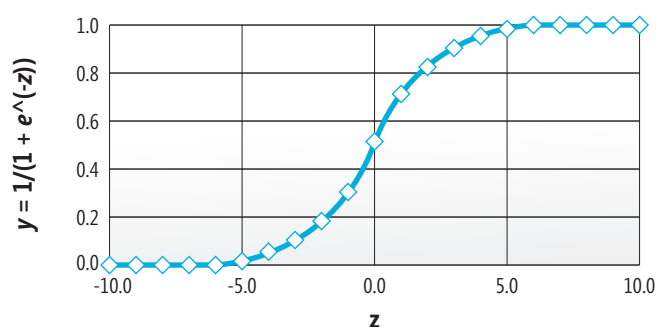


Figure 1 The Sigmoid Function

and use one of several techniques to find the values of beta that best fit the data. After you've determined the values of beta, you can use them to make predictions on new data. One of the most common techniques for finding the beta values for a logistic regression equation is called iteratively reweighted least squares (IRLS). IRLS starts with an estimate of the beta values and then iteratively computes a new, better set of betas until some stopping condition is met. There are several techniques that can be used to determine a new, better set of beta values. One of the most common is called Newton-Raphson (NR), which involves finding the calculus derivative of a function—in this case the derivative of the sigmoid function. Because of the close connection between IRLS and Newton-Raphson in logistic regression, the two terms are often used interchangeably.

Although there are plenty of resources available that describe the complex mathematics behind finding logistic regression beta parameters using Newton-Raphson, very few, if any, implementation guides for programmers exist. This article explains exactly how logistic regression with Newton-Raphson works, and how to implement a solution using the C# language. Take a look at the screenshot in **Figure 2** to see where I'm headed.

The demo program begins by generating two synthetic data files. The first is called the training file and consists of 80 lines of age, sex, cholesterol and death data. The training file is used to compute the LR beta values. The second is called the test file and holds 20 lines of data that's used to evaluate the accuracy of the LR equation with the beta values computed from the training data. The demo program loads the X predictor values of the training data into a matrix, and loads the Y dependent variable values of the data into a vector. Notice that the X training matrix, often called a design matrix, has an additional first column that consists of all 1.0 values, and that the

Code download available at archive.msdn.microsoft.com/mag201209TestRun.

predictor values have all been converted to numeric values. Next, the demo program sets three stopping conditions for the IRLS algorithm, indicated by variables maxIterations, epsilon and jumpFactor. The demo program uses the Newton-Raphson algorithm to estimate the b0, b1, b2 and b3 beta values that best fit the training data. The demo concludes by evaluating how accurate the resulting LR equation with the computed beta values is on the test data. In this example, 18 out of 20 Y values (90 percent) were correctly predicted.

This article assumes you have advanced programming skills and at least an intermediate knowledge of matrix operations and terminology, but doesn't assume you know anything about logistic regression. The code that produced the screenshot in **Figure 2** is far too large to present in its entirety here, but the complete source code is available at archive.msdn.microsoft.com/mag201208TestRun. Because of the complexity of the IRLS/NR algorithm, I'll focus primarily on key parts of the algorithm rather than on the code itself, so you'll be able to modify the code to meet your own needs or refactor it to another programming language if you wish.

Overall Program Structure

For simplicity, all the code that generated the screenshot in **Figure 2** is contained in a single C# console application. The program structure and Main method, with some WriteLine statements removed, are listed in **Figure 3**.

```

file:///C:/LogisticRegressionNewtonRaphson/bin/Debug/LogisticRegressionNewtonRaphson.EXE
Begin Logistic Regression with Newton-Raphson demo
Creating 80 lines of synthetic training data and 20 lines of test data
First 5 lines of training data file are:
Age Sex Chol Died
48 F 4.40 0
60 M 7.89 1
51 M 3.48 0
66 M 8.41 1
40 F 3.05 0
. . .
Loading train and test data from files into memory
First five rows of x training matrix:
1.0 48.0 1.0 4.4
1.0 60.0 0.0 7.9
1.0 51.0 0.0 3.5
1.0 66.0 0.0 8.4
1.0 40.0 1.0 3.1
First five rows of y training vector:
0.0
1.0
0.0
1.0
0.0
Setting Newton-Raphson algorithm stop conditions
maxIterations = 25 epsilon = 0.01 jumpFactor = 1000
Using Newton-Raphson to find beta parameters that best fit the training data
Newton-Raphson complete
The beta vector is:
-593.5820
2.1618
-17.9056
73.1443
Computing accuracy on test data using the beta values
The predictive accuracy of the model on the test data is 90.00%
End demo

```

Figure 2 Logistic Regression with Newton-Raphson

Although logistic regression is a complex topic, the code in **Figure 3** is not quite as complicated as it might first appear because most of the methods shown are relatively short helper routines. The two key methods are ComputeBestBeta and ConstructNewBetaVector.

The heart of logistic regression with Newton-Raphson is a routine that computes a new, presumably better, set of beta values from the current set of values.

The MakeRawData method generates a file of quasi-random age-sex-cholesterol-death data. Age is a random integer value between 35 and 80, sex is either M or F with equal probability, and cholesterol is a semi-random real value between 0.1 and 9.9 that's based on the current age value. The death dependent variable is computed using a logistic regression equation with fixed beta values of b0 = -95.0, b1 = 0.4, b2 = -0.9 and b3 = 11.2. So MakeRawData generates data that can definitely be modeled using LR, as opposed to generating purely random data that would likely not follow an LR model.

Computing a New Beta Vector

The heart of logistic regression with Newton-Raphson is a routine that computes a new, presumably better, set of beta values from the current set of values. The math is very deep, but fortunately the net result is not too complex. In pseudo-equation form, the update process is given by:

$$b[t] = b[t-1] + \text{inv}(X'W[t-1]X)(Y - p[t-1])$$

Here b[t] is the new ("at time t," not array indexing) beta vector. On the right-hand side, b[t-1] is the old ("at time t-1") beta vector. The inv function is matrix inversion. Uppercase X is the design matrix—that is, the values of the predictor variables augmented with a leading column of 1.0s. Uppercase X' is the transpose of the X design matrix. Uppercase Y is the vector of dependent variable values (recall each value will be 0 or 1). The quantity p[t-1] represents the vector of old predicted probability values for Y (which will consist of values between 0.0 and 1.0). The uppercase W quantity is a so-called weights matrix, which requires a bit of explanation.

The beta update equation and the W matrix are best explained with a concrete example. Suppose for simplicity that the training set

Figure 3 Program Structure

```
using System;
using System.IO;

namespace LogisticRegressionNewtonRaphson
{
    class LogisticRegressionNRProgram
    {
        static void Main(string[] args)
        {
            try
            {
                string trainFile = "trainFile.txt";
                string testFile = "testFile.txt";

                MakeRawDataFile(80, 3, trainFile);
                MakeRawDataFile(20, 4, testFile);

                Console.WriteLine("First 5 lines of training data file are: \n");
                DisplayRawData(trainFile, 5);

                double[][] xTrainMatrix = LoadRawDataIntoDesignMatrix(trainFile);
                double[] yTrainVector = LoadRawDataIntoYVector(trainFile);

                double[][] xTestMatrix = LoadRawDataIntoDesignMatrix(testFile);
                double[] yTestVector = LoadRawDataIntoYVector(testFile);

                int maxIterations = 25;
                double epsilon = 0.01;
                double jumpFactor = 1000.0;
                double[] beta = ComputeBestBeta(xTrainMatrix, yTrainVector,
                    maxIterations, epsilon, jumpFactor);
                Console.WriteLine("Newton-Raphson complete");
                Console.WriteLine("The beta vector is: ");
                Console.WriteLine(VectorAsString(beta, int.MaxValue, 4, 10));

                double acc = PredictiveAccuracy(xTestMatrix, yTestVector, beta);
                Console.WriteLine("The predictive accuracy of the model on the test
                    data is " + acc.ToString("F2") + "%\n");
            }
            catch (Exception ex)
            {
                Console.WriteLine("Fatal: " + ex.Message);
                Console.ReadLine();
            }
        } // Main

        static void MakeRawDataFile(int numLines, int seed, string fileName)

        static void DisplayRawData(string fileName, int numLines)

        static double[][] LoadRawDataIntoDesignMatrix(string rawDataFile)

        static double[] LoadRawDataIntoYVector(string rawDataFile)

        static double PredictiveAccuracy(double[][] xMatrix,
            double[] yVector, double[] bVector)

        static double[] ComputeBestBeta(double[][] xMatrix, double[] yVector,
            int maxNewtonIterations, double epsilon, double jumpFactor)

        static double[] ConstructNewBetaVector(double[] oldBetaVector,
            double[][] xMatrix,
            double[] yVector, double[] oldProbVector)

        static double[][] ComputeXtilde(double[] pVector, double[][] xMatrix)

        static bool NoChange(double[] oldBvector, double[] newBvector, double epsilon)

        static bool OutOfControl(double[] oldBvector, double[] newBvector,
            double jumpFactor)

        static double[] ConstructProbVector(double[][] xMatrix, double[] bVector)

        // Matrix and vector routines:

        static double MeanSquaredError(double[] pVector, double[] yVector)

        static double[][] MatrixCreate(int rows, int cols)

        static double[] VectorCreate(int rows)

        static string MatrixAsString(double[][] matrix, int numRows,
            int digits, int width)

        static double[][] MatrixDuplicate(double[][] matrix)

        static double[] VectorAddition(double[] vectorA, double[] vectorB)

        static double[] VectorSubtraction(double[] vectorA, double[] vectorB)

        static string VectorAsString(double[] vector, int count, int digits, int width)

        static double[] VectorDuplicate(double[] vector)

        static double[][] MatrixTranspose(double[][] matrix) // assumes
            matrix is not null

        static double[][] MatrixProduct(double[][] matrixA, double[][] matrixB)

        static double[] MatrixVectorProduct(double[][] matrix, double[] vector)

        static double[][] MatrixInverse(double[][] matrix)

        static double[] HelperSolve(double[][] luMatrix, double[] b) // helper

        static double[][] MatrixDecompose(double[][] matrix, out int[] perm,
            out int tog)

    } // Program
} // ns
```

consists only of the first five lines of data shown in **Figure 1**. So the design matrix X would be:

```
1.00  48.00  1.00  4.40
1.00  60.00  0.00  7.89
1.00  51.00  0.00  3.48
1.00  66.00  0.00  8.41
1.00  40.00  1.00  3.05
```

The Y dependent variable vector would be:

```
0
1
0
1
0
```

Let's assume that the old beta vector of values to be updated, $b[t-1]$, is:

```
1.00
0.01
0.01
0.01
```

With these values for X and beta, the old p vector, $p[t-1]$, is:

```
0.8226
0.8428
0.8242
0.8512
0.8085
```

Notice that if we presume p values < 0.5 are interpreted as $y = 0$ and p values ≥ 0.5 are interpreted as $y = 1$, the old beta values would correctly predict only two out of five cases in the training data.

The weights matrix W is an $m \times m$ matrix where m is the number of rows of X. All the values in the W matrix are 0.0 except for those m values that are on the main diagonal. Each of these values equals the corresponding p value multiplied by $1-p$. So for this example, W would be size 5×5 . The upper-left cell at $[0,0]$ would be $(0.8226)(1 - 0.8226) = 0.1459$. The cell at $[1,1]$ would be $(0.8428)(1 - 0.8428) = 0.1325$, and so on. The quantity $(p)(1-p)$ represents the calculus derivative of the sigmoid function.

In practice, the W matrix is not computed explicitly because its size could be huge. If you had 1,000 rows of training data, matrix W would have 1,000,000 cells. Notice the beta update equation has a term $W[t-1]X$, which means the matrix product of $W[t-1]$ and X . Because most of the values in $W[t-1]$ are zero, most of the matrix multiplication terms are also zero. This allows $W[t-1]$ times X to be computed directly from $p[t-1]$ and X , without explicitly constructing W . Several of the math references that describe IRLS with the NR algorithm for LR use the symbol $X\sim$ (X tilde) for the product of $W[t-1]$ and X . See method `ComputeXtilde` in the code download for implementation details.

Method `ConstructNewBetaVector` accepts as input parameters the old beta vector, the X design matrix, the Y dependent variable vector and the old probability vector. The method computes and returns the updated beta vector.

The method is implemented like so:

```
double[][] Xt = MatrixTranspose(xMatrix); // X'
double[][] A = ComputeXtilde(oldProbVector, xMatrix); // WX
double[][] B = MatrixProduct(Xt, A); // X'WX
double[][] C = MatrixInverse(B); // inv(X'WX)
if (C == null) // inverse failed!
    return null;
double[][] D = MatrixProduct(C, Xt); // inv(X'WX)'
double[] YP = VectorSubtraction(yVector, oldProbVector); // Y-p
double[] E = MatrixVectorProduct(D, YP); // inv(X'WX)X'(y-p)
double[] result = VectorAddition(oldBetaVector, E);
return result;
```

With the collection of matrix and vector helper methods shown in **Figure 3**, computing a new beta vector is fairly straightforward. Note that the method performs matrix inversion. This is a process that can go wrong in many ways and is a significant weakness of Newton-Raphson.

Continuing the example, matrix Xt (the transpose of X) would be:

1.00	1.00	1.00	1.00	1.00
48.00	60.00	51.00	66.00	40.00
1.00	0.00	0.00	0.00	1.00
4.40	7.89	3.48	8.41	3.05

Matrix A ($X\sim$) would be computed from vector p and matrix X by helper method `ComputeXtilde` as:

0.15	7.00	0.15	0.64
0.13	7.95	0.00	1.05
0.14	7.39	0.00	0.50
0.13	8.36	0.00	1.07
0.15	6.19	0.15	0.47

Intermediate matrix B , representing the product of Xt and $X\sim$ (which in turn is $XtW[t-1]X$) would be:

0.70	36.90	0.30	3.73
36.90	1989.62	13.20	208.46
0.30	13.20	0.30	1.11
3.73	208.46	1.11	23.23

Intermediate matrix C is the inverse of matrix B and would be:

602.81	-14.43	-110.41	38.05
-14.43	0.36	2.48	-1.02
-110.41	2.48	26.43	-5.77
38.05	-1.02	-5.77	3.36

Intermediate matrix D is the product of matrix C and matrix $X\sim$ transpose and would be computed as:

-33.00	37.01	-0.90	-29.80	31.10
0.77	-0.96	0.30	0.66	-0.72
9.52	-7.32	-4.17	4.54	-2.51
-1.86	3.39	-2.24	-0.98	1.76

Intermediate vector YP is the difference between vectors Y and $p[t-1]$ and would be:

-0.8
0.2
-0.8
0.1
-0.8

Intermediate vector E is the product of matrix D and vector YP and holds the increments to be added to the old beta vector. Vector E would be:

4.1
-0.4
-2.8
2.3

The new, final beta vector is obtained by adding the values in intermediate vector E to the old values of beta, and in this example would be:

5.1
-0.3
-2.8
2.4

With the new values of beta, the new values for the p vector would be:

0.0240
0.9627
0.0168
0.9192
0.0154

Figure 4 Computing the Best Beta Vector

```
static double[] ComputeBestBeta(double[][] xMatrix, double[] yVector,
    int maxIterations, double epsilon, double jumpFactor)
{
    int xRows = xMatrix.Length;
    int xCols = xMatrix[0].Length;
    if (xRows != yVector.Length)
        throw new Exception("xxx (error)");

    // Initialize beta values
    double[] bVector = new double[xCols];
    for (int i = 0; i < xCols; ++i) { bVector[i] = 0.0; }
    double[] bestBVector = VectorDuplicate(bVector);

    double[] pVector = ConstructProbVector(xMatrix, bVector);
    double mse = MeanSquaredError(pVector, yVector);
    int timesWorse = 0;

    for (int i = 0; i < maxIterations; ++i)
    {
        double[] newBVector = ConstructNewBetaVector(bVector, xMatrix,
            yVector, pVector);
        if (newBVector == null)
            return bestBVector;

        if (NoChange(bVector, newBVector, epsilon) == true)
            return bestBVector;

        if (OutOfControl(bVector, newBVector, jumpFactor) == true)
            return bestBVector;

        pVector = ConstructProbVector(xMatrix, newBVector);

        double newMSE = MeanSquaredError(pVector, yVector); // Smaller is better
        if (newMSE > mse) // new MSE is worse
        {
            ++timesWorse; // Update got-worse counter
            if (timesWorse >= 4)
                return bestBVector;

            bVector = VectorDuplicate(newBVector); // Attempt escape
            for (int k = 0; k < bVector.Length; ++k)
                bVector[k] = (bVector[k] + newBVector[k]) / 2.0;
            mse = newMSE;
        }
        else // Improved
        {
            bVector = VectorDuplicate(newBVector);
            bestBVector = VectorDuplicate(bVector);
            mse = newMSE;
            timesWorse = 0;
        }
    } // End main iteration loop
    return bestBVector;
}
```

If these p values are interpreted as $Y = 0$ when $p < 0.5$, then, after only one iteration of Newton-Raphson, the beta values correctly predict all five cases in the test data.

Knowing When to Stop

The Newton-Raphson technique for logistic regression iteratively improves the values of the beta vector until some stopping condition is met. It's surprisingly difficult to know when to stop the iteration. Method `ComputeBestBeta` handles this task. The code is presented in **Figure 4**.

One of the biggest pitfalls of logistic regression is iterating too many times, resulting in a set of beta values that fit the training data perfectly but don't fit any other data sets well. This is called over-fitting. Knowing when to stop the training process in logistic regression is part art and part science. Method `ComputeBestBeta` begins by initializing the beta vector to all 0.0 values, computing the associated p values, and then computing the error between the p values and the Y values. The code in this article uses mean squared error—the average of the sums of squared differences between p and Y . Other possibilities for computing error include average absolute deviation and cross-entropy error.

The main processing loop in `ComputeBestBeta` repeatedly computes a new beta vector and corresponding error term. The primary stopping condition in the algorithm is a `maxIterations` parameter. Recall that the Newton-Raphson algorithm computes a matrix inverse, which is susceptible to failing. In this case `ComputeBestBeta` returns the best beta vector found (which, unfortunately, could be the initial all-zero vector). You might want to throw an exception here instead. Another alternative is to attempt an escape by modifying the new beta values by adjusting them to the average of the previous values and the new values.

One of the biggest pitfalls of
logistic regression is iterating too
many times.

The next stop condition checks to see if the change in all the new beta values is smaller than some small value of parameter `epsilon`, using helper method `NoChange`. This indicates that Newton-Raphson has converged and, in fact, there's a good chance your model is now over-fitted. Instead of returning the best beta vector found at this point, you might want to return the best beta vector from an earlier iteration. The next stop condition checks to see if any of the new beta values have jumped by an amount greater than some large value given by parameter `jumpFactor`. This indicates that Newton-Raphson has possibly spun out of control and you want to throw an exception instead of retuning the best beta vector found.

Another stop condition in `ComputeBestBeta` checks to see if the new beta values actually produce a larger error than the previous beta values did. In this example, if new betas produce a larger error four times in a row, the algorithm terminates and returns the best beta vector found. You might want to parameterize the

value for the maximum number of consecutive increases in error. When an increase in error is detected, the method attempts to escape the situation by changing the values in the current beta vector to the average of the values in the current beta and the values in the newly computed beta.

There's no single best set of stopping conditions. Every logistic regression problem requires a bit of experimentation to tune the Newton-Raphson algorithm.

Knowing when to stop
the training process in logistic
regression is part art and
part science.

Advantages vs. Disadvantages

There are several alternatives to Newton-Raphson for estimating the best set of beta values in logistic regression. Newton-Raphson is a deterministic numerical-optimization technique. You can also employ non-deterministic (meaning probabilistic) techniques such as particle swarm optimization, evolutionary algorithms and bacterial foraging optimization.

The primary advantage of Newton-Raphson compared to probabilistic alternatives is that in most situations NR is much faster. But Newton-Raphson has several disadvantages. Because NR uses matrix inversion, the algorithm will fail when encountering a singular matrix during computation. Simple implementations of Newton-Raphson require all data to be in memory, which means there's a limit to the size of the training and test data sets you can use. Although it's rare, some datasets might not converge to a set of best beta values at all using NR.

When I use logistic regression, I often employ a two-pronged attack. I first experiment with a Newton-Raphson approach. If that technique fails, I fall back to using particle swarm optimization to find the best set of beta values. It's important to note that logistic regression isn't magic, and not all data fits a logistic regression model. Other machine-learning techniques to model data with a binary-dependent variable include neural networks, support vector machines and linear-discriminant analysis.

The explanation of the beta vector update process of the Newton-Raphson algorithm presented in this article and the accompanying code download should get you up and running with logistic regression using NR. Logistic regression is a fascinating, complex topic and can make a valuable addition to your personal skill set. ■

DR. JAMES McCaffrey works for Volt Information Sciences Inc., where he manages technical training for software engineers working at the Microsoft Redmond, Wash., campus. He has worked on several Microsoft products including Internet Explorer and MSN Search. McCaffrey is the author of "NET Test Automation Recipes" (Apress, 2006). He can be reached at jammc@microsoft.com.

THANKS to the following technical experts for reviewing this article:
Dan Liebling and Anne Loomis Thompson



PLATINUM SPONSOR



GOLD SPONSOR

Metalogix

SharePoint Live Virtual Conference & Expo offers in-depth technical training for IT professionals and developers. Learn best practices from the field as our technical experts show you how to deploy, manage and build solutions for Microsoft SharePoint.

Keynote Address:

Steve Fox, Director in MCS, Microsoft

Top Sessions Include:

- Automating SharePoint Governance and Management
- Implementing SharePoint 2013 ECM Solutions
- How We Built Our App for SharePoint 2013 Marketplace
- What's New with Web Content Management in SharePoint 2013

Register today for this FREE virtual event: SharePointVCX.com

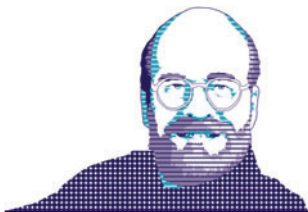
PRODUCED BY



Microsoft | TechNet

Visual Studio
MAGAZINE





Exploring Spherical Coordinates on Windows Phone

Ever since I became acquainted with the Motion sensor in Windows Phone 7.1, I've known exactly what I wanted to do with it: resurrect some C# positional astronomy code I wrote about five years ago and wire it up with Motion sensor logic into a Windows Phone program. Such a program would allow me to point the phone at a spot in the night sky to show the planets and constellations located there.

That program—with the ridiculous name AstroPhone—is coming in the next installment of this column. Meanwhile, some conceptual, mathematical and programmatic groundwork is necessary.

The Motion sensor consolidates
input from the compass,
accelerometer, gyroscope and
GPS, and computes a 3D rotation
matrix that describes the phone's
orientation in space.

As I've been discussing in previous columns, the Motion sensor consolidates input from the compass, accelerometer, gyroscope and GPS, and computes (among other things) a 3D rotation matrix that describes the phone's orientation in space. It's convenient to apply this matrix to a 3D vector relative to the phone's coordinate system. This transformed vector points to a location on the conceptual celestial sphere. A location on this sphere is a horizontal coordinate, which consists of an altitude—an angle above or below the viewer's horizon—and an azimuth, which is basically a compass direction.

Horizontal coordinates are analogous to the familiar geographic coordinates that identify a position on the surface of the Earth. The altitude above or below the viewer's horizon is analogous to a latitude above or below the Earth's equator. The azimuth indicating a compass direction around the viewer's horizon is analogous to the longitude around the Earth's equator. As you'll see, all types

of spherical coordinates are fundamentally the same, differing most crucially by the plane that separates the upper hemisphere from the lower.

Flattening the Sphere

In last month's column (msdn.microsoft.com/magazine/jj553520), I demonstrated how to use horizontal coordinates to scan large photographs and other images by changing the phone's orientation in space. However, that program “cheated” a bit by simply ignoring the true nature of spherical coordinates. It refused to acknowledge the fact that circles of latitude become increasingly smaller toward the top and bottom of the celestial sphere, and consequently, degrees of azimuth become increasingly compressed.

A program that provides a simulated viewing of the night sky simply can't take those shortcuts. It must find an algorithmic way to map an area of the celestial sphere to the flat screen of the phone. Of course, flattening the sphere has been a problem encountered by mapmakers for centuries. It can't be done without distortion, but the objective is to minimize those distortions.

Figure 1 shows a sphere decorated with lines of altitude and azimuth every 15°. From a vantage point at the center of the inside of the sphere, the red dot represents a point that you're viewing or pointing at with your phone. The altitude of this particular point is 25°. The azimuth depends on where you start counting and in what direction.

Let's call this point the “view center” because it's the point that will be mapped to the center of the phone's screen. What lines on the sphere correspond to the edges of the screen?

It's tempting to assume that the edges of the phone's screen will correspond to lines of altitude and azimuth. But that only seems reasonable at locations near the horizon. The concept starts to break down as the view center gets further from the horizon, and collapses altogether when you're viewing straight up or down at the poles.

Instead, let's construct orthogonal “great circles” at the view center, as shown in **Figure 2**.

Great circles are an important concept in spherical geometry. A great circle is a two-dimensional circle in 3D space sharing the same radius and center as the sphere itself. Great circles on the sphere are analogous to straight lines on a plane because they represent the shortest distance between two points. Lines of azimuth (and longitude) are great circles. Lines of altitude (and latitude) are not; they are instead “small circles,” except for the circle that divides the sphere in two—the horizon (or equator).

Code download available at archive.msdn.microsoft.com/mag201209TouchAndGo.

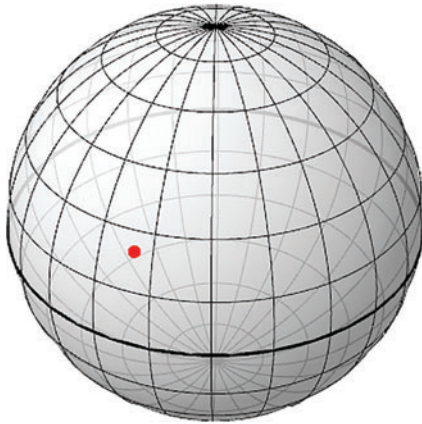


Figure 1 A Point (in Red) on a Celestial Sphere

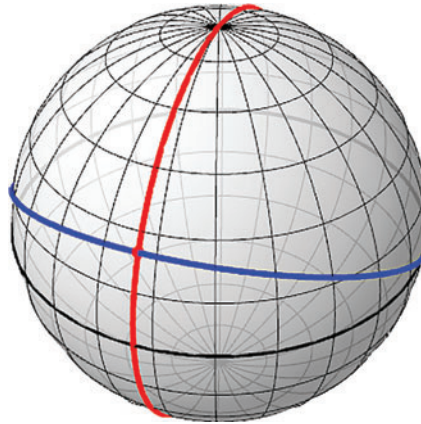


Figure 2 Orthogonal Great Circles Crossing the View Center

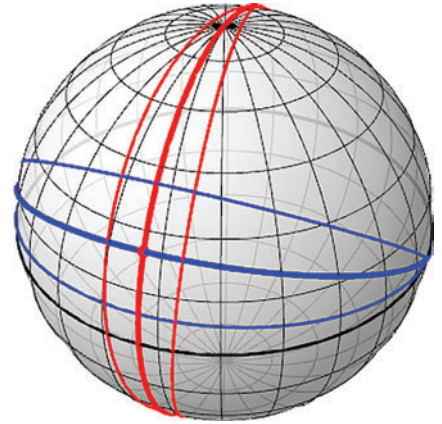


Figure 3 Additional Great Circles to Create a Rectangular Area

The red great circle in **Figure 2** passes through the two poles and is simply an azimuth line. The blue great circle crosses the horizon at azimuths 90° from the azimuth of the view center.

A rectangle surrounding that view center will be mapped to the phone's screen. Conceptually, this rectangle can be a grid of great circles. Let's construct two more great circles in blue that cross the horizon at the same point as the first one. And let's construct two more red great circles that also share two points with the first one. The result is shown in **Figure 3**. The shared points of the red great circles have the same azimuth as the view center (or the azimuth plus 180°) but the altitude is offset by 90°.

Those additional great circles define the edges of a rectangle that corresponds to the portrait-mode screen of the phone, with a width of about 19° of arc and a height of 32° of arc. If you assume that the phone's screen is about 2 inches wide and 3.33 inches high, these arc degrees are appropriate when the phone's screen is held about 6 inches from your face. The phone's screen is 480 pixels wide and 800 pixels high, so these numbers also imply 25 pixels per arc degree.

When implementing this projection scheme in an algorithm, you need to work backward: Think of the two points shared by the three blue great circles as an axis, and the two points shared by the three red great circles as another axis. Together with the vector to the view center, these form an orthonormal coordinate system.

A great circle is a two-dimensional circle in 3D space sharing the same radius and center as the sphere itself.

Vector algebra allows deriving angle offsets of any other coordinate from the view center, as shown in the `HorizontalCoordinateProjection` class in **Figure 4**. The class has one method to set the view center, and another method to obtain angle offsets of the horizontal coordinate of any other object relative to that view center.

To determine where that object should be positioned on the phone's screen, the angles obtained from the `GetAngleOffsets` method need to be multiplied by a constant you select for the number of pixels per arc degree. I suggested earlier that this constant should equal 25 when the phone is held 6 inches from your face, but you might want to go with something lower to provide a broader view.

Viewing the Sphere from the Inside

The `ViewHorizontalCoordinates` program sets its `PIXELS_PER_DEGREE` constant to 15 to display horizontal coordinates from the inside looking out, as shown in **Figure 5**. That particular view occurs when the phone points a little south of east, a bit above the horizon.

For each `CurrentValueChanged` event fired by the Motion sensor, the event handler begins by obtaining the rotation matrix that indicates how the Earth is oriented relative to the phone. It converts

Figure 4 The `HorizontalCoordinateProjection` Class

```
public class HorizontalCoordinateProjection
{
    Vector3 viewCenterVector, horzAxis, vertAxis;

    public void SetViewCenter(HorizontalCoordinate viewCenterCoord)
    {
        viewCenterVector = viewCenterCoord.ToVector();
        HorizontalCoordinate vertAxisCoord =
            new HorizontalCoordinate(viewCenterCoord.Azimuth + 90, 0);
        vertAxis = vertAxisCoord.ToVector();
        horzAxis = Vector3.Cross(viewCenterVector, vertAxis);
    }

    public void GetAngleOffsets(HorizontalCoordinate objectCoord,
        out double horzAngle, out double vertAngle)
    {
        Vector3 objectVector = objectCoord.ToVector();
        Vector3 horzObjectCross = Vector3.Cross(objectVector, -horzAxis);
        Vector3 vertObjectCross = Vector3.Cross(objectVector, vertAxis);

        horzObjectCross.Normalize();
        vertObjectCross.Normalize();

        double x = Vector3.Dot(horzObjectCross, vertAxis);
        double y = Vector3.Dot(horzObjectCross, viewCenterVector);
        horzAngle = -180 * Math.Atan2(y, x) / Math.PI;

        x = Vector3.Dot(vertObjectCross, horzAxis);
        y = Vector3.Dot(vertObjectCross, viewCenterVector);
        vertAngle = -180 * Math.Atan2(y, x) / Math.PI;
    }
}
```

that to a `HorizontalCoordinate` value, and sets the view center in a previously created `HorizontalCoordinateProjection` object:

```
Microsoft.Xna.Framework.Matrix matrix =
    args.SensorReading.Attitude.RotationMatrix;
HorizontalCoordinate viewCenter =
    HorizontalCoordinate.FromMotionMatrix(matrix);
coordinateProjection.SetViewCenter(viewCenter);
rotate.Angle = -viewCenter.Tilt;
```

The `rotate` object is a `RotateTransform` applied to the entire `Canvas`. The handler then implements several loops involving altitude and azimuth values in 15° increments. The first time the `CurrentValueChanged` event fires, the event handler creates all the necessary `Line` and `TextBlock` elements and adds them to the `Canvas`. The second and subsequent times through, the handler simply accesses the existing `Line` and `TextBlock` elements already in the `Canvas` and sets new points.

Each `HorizontalCoordinate` value needs to be converted to a screen coordinate. That's the job of the `CalculateScreenPoint` method in **Figure 6**, which calls the `GetAngleOffsets` method in `HorizontalCoordinateProjection` and multiplies the angles by the `PIXELS_PER_DEGREE` constant.

The `GetAngleOffsets` method always returns angles in the range -180° to 180° . Sometimes lines straddle these limits and this creates a wraparound problem. For example, a line might (in theory) extend from -175° to 175° . That line should be only 10° in length, but the calculated length would be 380° ! The logic in `CalculateScreenPoint` involving `NaN` ("not a number") corrects this problem by flagging all points with angle offsets less than -90° or greater than 90° , which is out of the range of the screen.

I wanted the text labels showing the altitude angles to be visible regardless of the azimuth, and those showing the compass points to be visible no matter how high or low you point the phone. The altitude labels are displayed with a screen point calculated from the view center azimuth, and the compass labels are displayed with a screen point calculated from the view center altitude, with a little adjustment so they don't all cluster together when you point the phone straight up or down. This little trick helps keep the labels inside the screen, perhaps rotated a bit away from the center.

Figure 6 Calculating a Screen Point from a Horizontal Coordinate

```
Point CalculateScreenPoint(HorizontalCoordinate horizontalCoordinate)
{
    double horzAngle, vertAngle;
    coordinateProjection.GetAngleOffsets(horizontalCoordinate,
                                         out horzAngle, out vertAngle);

    // Use NaN to indicate points clearly out of range of the screen
    float x = float.NaN;
    float y = float.NaN;

    if (horzAngle > -90 && horzAngle < 90 && vertAngle > -90 && vertAngle < 90)
    {
        x = (float)(width / 2 + PIXELS_PER_DEGREE * horzAngle);
        y = (float)(height / 2 + PIXELS_PER_DEGREE * vertAngle);
    }
    return new Point(x, y);
}
```

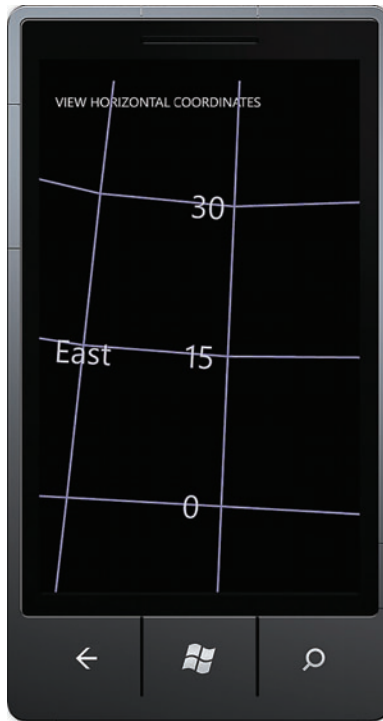


Figure 5 The ViewHorizontalCoordinates Program

A Switch to XNA

As I was working on the programs for this column in conjunction with `AstroPhone` for next month, I started to notice some performance issues. The `ViewHorizontalCoordinates` program can only manage about 10 frames per second on my development phone, and the problem seems to be more in the Silverlight layout system rather than in my code.

With some reluctance I decided that the remaining programs would target the Windows Phone XNA framework rather than Silverlight. This is the case for the `ViewThreeCoordinates` project, which is similar to `ViewHorizontalCoordinates` but written for XNA rather than Silverlight.

Library Preview

The `ViewThreeCoordinates` project also reveals some of the strategies I'll be using for the full-blown astronomy program. It makes use of part of a larger library called `Petzold.Astronomy`. In constructing this library, my primary reference has been the second edition of the classic book "Astronomical Algorithms" by Jean Meeus (Willmann-Bell, 1998).

The plane of the Earth's equator is the fundamental plane associated with equatorial coordinates.

Positional astronomy requires a lot of trigonometry. In the `Petzold.Astronomy` library, I try to avoid confusion and conversions between degrees and radians by implementing a struct named `Angle`. You can create an `Angle` value from either a degree or a radian using the static `FromDegree` or `FromRadian` methods, and get the degrees or radians out, but it's not often necessary because `Angle` also implements all the necessary trigonometric functions as properties.

For example, if you have an `Angle` value named `myAngle`, you can obtain the cosine of that angle like so:

```
double x = myAngle.Cosine;
```

If you think of sines, cosines and tangents as "properties" of angles, this makes perfect sense. But notice how the `Cosine` property is implemented:

```
public double Cosine
{
    get { return Math.Cos(radians); }
    set { radians = Math.Acos(value); }
}
```

The `set` accessor is the inverse cosine function, which means you can set an existing `Angle` value to the inverse cosine of a number using a statement like this:

```
myAngle.Cosine = x;
```


The only place where this process didn't quite work was in implementing the essential `Atan2` method. I did it with a static method that creates an `Angle` value:

```
public static Angle ArcTangent(double y, double x)
{
    return Angle.FromRadians(Math.Atan2(y, x));
}
```

Equatorial Coordinates

The `ViewThreeCoordinates` program displays horizontal coordinates in green, plus two other coordinate systems very useful in positional astronomy: equatorial coordinates in blue and ecliptic coordinates in red. **Figure 7** shows a typical display with the phone pointing north and an altitude of 25°. (Yes, I know the screen is cluttered, but you can tap the screen to display only one or two sets of coordinates.)

As the Earth rotates on its axis and revolves around the sun, the Earth's equator stays pretty much in the same plane. The plane of the Earth's equator is the fundamental plane associated with equatorial coordinates. Equatorial coordinates consist of a declination—positive above the equator and negative below—and a right ascension going around the equator. The right ascension is usually specified in hours rather than degrees, where each hour is equivalent to 15°.

Of course, geographic coordinates (longitude and latitude) are also based on the Earth's equator, but geographic coordinates rotate with the Earth while equatorial coordinates are fixed relative to the rest of the universe, and hence seem to turn as the Earth rotates on its axis. The positions of stars are specified in equatorial coordinates, and they don't change much with the passage of time.

The orbits of all the planets
of the solar system are roughly in
the same plane.

`ViewThreeCoordinates` displays a grid of equatorial coordinates by converting them to horizontal coordinates. This conversion depends on the observer's geographic location (longitude and latitude) and the current time, and is implemented in the `HorizontalCoordinate` structure.

However, a very rough conversion of right ascension to azimuth is possible: At a local time of midnight on the vernal equinox (March 20 or thereabouts), 0 hours right ascension is north (0° of azimuth). The bright star Arcturus has a right ascension of about 14 hours, so on that date at that time, you'll need to swing westward by 210° (or eastward by 150°) to see it. Or, you can just wait two hours until 2 a.m. and Arcturus will be due south.

You can calculate the local hour angle of a star by subtracting its right ascension from the number of hours since midnight local



Figure 7 The `ViewThreeCoordinates` Display

time. Convert to degrees by multiplying by 15, and add the number of days since March 21, and that's the azimuth of the star measured eastward from north.

If you run `ViewThreeCoordinates` and hold the phone up to the night sky, the northern equatorial pole (visible at the very top of **Figure 7**) will correspond with the location of Polaris, the North Star, which has a declination of very nearly 90° and hence coincides with the vector of the Earth's axis. Notice how the azimuth line of North intersects that pole.

Ecliptic Coordinates

The orbits of all the planets of the solar system are roughly in the same plane. This plane is called the ecliptic, and it's the basis of ecliptic coordinates (also known as celestial coordinates), which are displayed by `ViewThreeCoordinates` in red. An ecliptic coordinate consists of an ecliptic longitude and ecliptic latitude.

Ecliptic coordinates are mostly used when computing the positions of the sun, planets and moon. Because the sun and planets lie in roughly the same plane, the ecliptic latitude of these objects is usually close to zero. The ecliptic

itself is displayed with a thick red line in `ViewThreeCoordinates`. If you hold up the phone to the day or night sky, the sun and planets should coincide with that line.

Equatorial coordinates and ecliptic coordinates differ solely as a result of the Earth's tilt of about 23° relative to the ecliptic. An equatorial coordinate of 0 hours right ascension and 0° declination is the same as an equatorial coordinate of 0° longitude and 0° latitude, and the two systems also coincide at 180°.

On the vernal equinox, the sun has an ecliptic longitude of 0°. That longitude increases by approximately 1° per day over the course of the year. Hence, the ecliptic longitude of the sun in degrees is approximately equal to the number of days since March 21.

Ecliptic longitudes are often labeled with the signs of the zodiac, starting with Aries for 0°-30°, Taurus for 30°-60°, Gemini for 60°-90° and so forth through Pisces for 330°-360°. I have also done this in `ViewThreeCoordinates`. These signs of the zodiac are constellations that are approximately located at these ecliptic longitudes with ecliptic latitudes in the neighborhood of zero. Thus, the sun is "in Aries" (meaning that Aries is beyond the sun) during the month beginning March 21.

Enough theory. In the next appearance of this column, the grid lines of the spherical coordinate systems will be replaced with the sun, moon, planets, stars and constellations. ■

CHARLES PETZOLD is a longtime contributor to *MSDN Magazine*, and is currently updating his classic book *"Programming Windows"* (Microsoft Press, 1998) for Windows 8. His Web site is charlespetzold.com.

THANKS to the following technical expert for reviewing this article: *Donn Morse*



On Honor, Cold Iron and Hot Silicon

I remember, at age 6, attending my father's medical school graduation. His face glowed as he explained to me the Hippocratic oath. "We swore to use the things we learned never to hurt anyone; only to make them better," he explained. I swear I remember a catch in his voice, though perhaps the intervening half-century has added that.

I sat at my own graduation from Dartmouth 18 years later (indoors in the hockey rink because it was raining outside), with my Thayer School of Engineering classmates, sharing my pal's flask as the medical students took their oath: "I swear by Apollo the Physician ..." We didn't have any such ritual, being mere engineers, a lower calling held to a lower standard.

A guy who develops software
for a jumbo jet holds more lives
in his hand on a single afternoon
than a doctor does in her entire
career. It's time we had a public
reminder of this.

Our responsibilities have grown exponentially since the dawn of the software industry, perhaps 50 years ago, about when my Dad was taking his oath. A guy who develops software for a jumbo jet holds more lives in his hand on a single afternoon than a doctor does in her entire career. It's time we had a public reminder of this.

I don't mean some empty "Code of Ethics," which all professional societies seem to have. These carry about as much meaning and power as a Web site's privacy statement, or the vows of fidelity in a French marriage ceremony. I want a public, individual promise, on a person's honor, in front of peers and betters and loved ones.

I found that such a thing already exists. It started in Canada in 1922, and is known as the "Ritual of the Calling of an Engineer" (bit.ly/8XwJlQ). Rudyard Kipling wrote the oath. He was a huge fan of engineering, which is evident in his classic work "McAndrew's Hymn" (bit.ly/htBlTe). Instead of a dead Greek god, the participants swear "upon Honor and Cold Iron." The new engineer is given a steel ring to wear on the little finger of his working hand, as an

external symbol and an internal reminder. An American version of this ceremony exists as well—the "Order of the Engineer" (bit.ly/14g7Pn)—though it's not widespread. It begins with the words: "I am an Engineer. In my profession I take deep pride. To it I owe solemn obligations."

These are good places to start. But their historical roots in metal-bashing, which hark back to the first engineer perhaps 5,000 years ago, don't match today's software engineering. We share with all engineers the challenges of problem definition and analysis, of making tradeoffs between time and cost and safety. On the other hand, we don't produce anything tangible, nothing you can really drop on your foot. Our end products are strings of ones and zeros, binary integers, no less and no more. Our ceremony needs to reflect the non-tangibility of our product.

What do we swear by? If we want to go the dead Greek route like doctors, how about Pythagoras, the first geek? His eponymous theorem—and, more important, his concept of mathematical proof—have withstood the millennia a whole lot better than Hippocrates' prescription of willow bark for pain relief. Or if Pythagoras is too Western or too dead, how about remembering our end product? Perhaps this: "On my honor, by the holy opposites zero and one, I swear ..."

What do we wear to show our art? Rings are easily visible, but even an extra few grams on a seldom-used finger add up after a day on the keyboard and mouse. A piercing? Perhaps, but in what? (No, not *that*.) How about a bar code tattoo, or better yet, a QR code? Each would carry a GUID unique to the oath-taker. They'd be decipherable with the certifying agency's public encryption key, so we'd know they were genuine. Any potential employer could check, using a smartphone app.

Now that I've stirred up this wasp's nest, I want to hear from you. I'll select the best ideas for our oath. Eight clauses there will be—a power of two, of course. What do we promise to do, or refrain from doing? Tell me what you think by sending an e-mail to the magazine at mmeditor@microsoft.com. And not just, "I will not drink at lunch, unless I'm thirsty." ■

DAVID S. PLATT teaches programming .NET at Harvard University Extension School and at companies all over the world. He's the author of 11 programming books, including "Why Software Sucks" (Addison-Wesley Professional, 2006) and "Introducing Microsoft .NET" (Microsoft Press, 2002). Microsoft named him a Software Legend in 2002. He wonders whether he should tape down two of his daughter's fingers so she learns how to count in octal. You can contact him at rollthunder.com.

Total File Coverage



Aspose.Words

DOC, DOCX, RTF, HTML, PDF,
XPS & other document formats.

Aspose.Cells

XLS, XLSX, XLSM, XLTX, CSV,
SpreadsheetML & image formats.

Aspose.Pdf

PDF, XML, XLS-FO, HTML, BMP,
JPG, PNG & other image formats.

Aspose.Email

MSG, EML, PST, EMLX &
other formats.



and many more!

Aspose.BarCode Aspose.Tasks
Aspose.Slides Aspose.Diagram
Aspose.OCR Aspose.Imaging



Create
Modify
Convert
Combine
& Print



Follow us on
Facebook & Twitter



Scan our QR Code
for an exclusive
20% coupon code.



Get your FREE evaluation copy at <http://www.aspose.com>

US Sales: 1.888.277.6734
sales@aspose.com

EU Sales: +44 (0) 141 416 1112
sales.europe@aspose.com

AU Sales: +61 2 8003 5926
sales.asiapacific@aspose.com

Introducing

The *Succinctly* Series

by Syncfusion



- Learn from industry experts.
- Around 100 pages of dense information.
- Clear and concise examples.
- Kindle and PDF formats.



Download the e-books at syncfusion.com/ebooks

